DSC 450: Database Processing for Large-Scale Analytics

Take-home Final

Name: Nachiketh Paramahamsa Reddy
ID: 2117731

Part 1

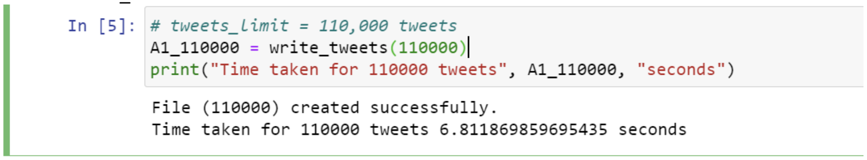
We will use one full day worth of tweets as our input (there are total of 4.4M tweets in this file, but we will intentionally use fewer tweets to run this final):
http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt
Execute and time the following tasks with 110,000 tweets and 550,000 tweets:

a. Use python to download tweets from the web and save to a local text file (not into a database yet, just to a text file). This is as simple as it sounds, all you need is a for-loop that reads lines from the web and writes them into a file.

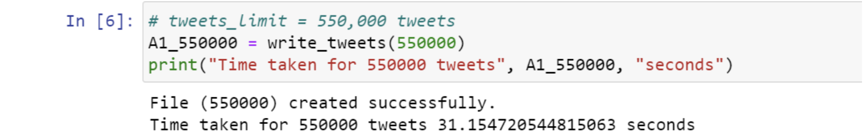
NOTE: Do not call read() or readlines(). That command will attempt to read the entire file which is too much data. Clicking on the link in the browser would cause the same problem.

CODE and O/P:

```
#1A
import urllib.request
import time

def write_tweets(tweets_limit):
    url = "http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"
    OP_File = f"tweets_{tweets_limit}.txt"

    start = time.time()
    count = 0
    with urllib.request.urlopen(url) as tweet_response:
            with open(OP_File, "w", encoding="utf-8") as file:
                for line in tweet_response:
                    file.write(line.decode().strip() + "\n")
                    count += 1

                    if count >= tweets_limit:
                        break

    end = time.time()
    time_taken = end - start

    print(f"File ({tweets_limit}) created successfully.")
    return time_taken
```

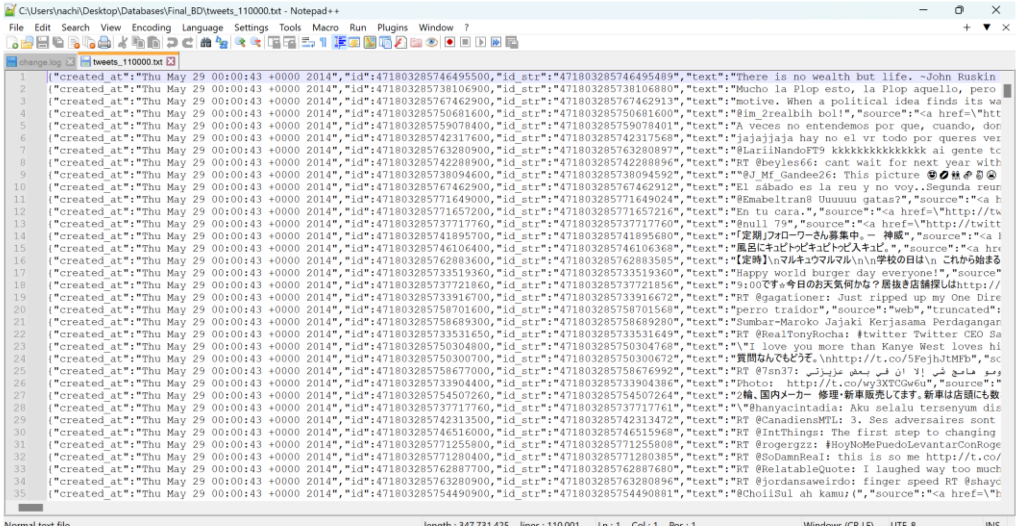


```
In [5]: # tweets_limit = 110,000 tweets
        A1_110000 = write_tweets(110000)
        print("Time taken for 110000 tweets", A1_110000, "seconds")

        File (110000) created successfully.
        Time taken for 110000 tweets 6.811869859695435 seconds
```



```
In [6]: # tweets_limit = 550,000 tweets
        A1_550000 = write_tweets(550000)
        print("Time taken for 550000 tweets", A1_550000, "seconds")

        File (550000) created successfully.
        Time taken for 550000 tweets 31.154720544815063 seconds
```

FILES:

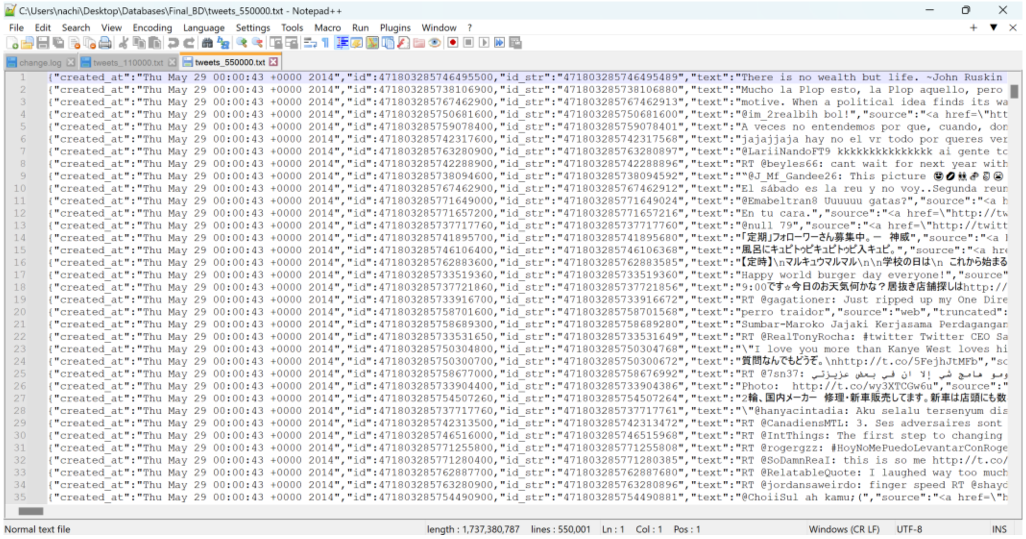


C:\Users\nachi\Desktop\Databases\Final_BD\tweets_110000.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
tweets_110000.txt

```
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746495500,"id_str":"471803285746495489","text":"There is no wealth but life. ~John Ruskin
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285738106900,"id_str":"471803285738106880","text":"Mucho la Plop esto, la Plop aquello, pero
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285767462900,"id_str":"471803285767462913","text":"motive. When a political idea finds its wa
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750681600,"id_str":"471803285750681600","text":"@im_2realbih bol!","source":"<a href=\"htt
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285759078400,"id_str":"471803285759078401","text":"A veces no entendemos por que, cuando, don
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742317600,"id_str":"471803285742317568","text":"jajajjaja hay no el vr todo por queres ver
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285763280900,"id_str":"471803285763280897","text":"@LariiNandoFT9 kkkkkkkkkkkkkk ai gente tc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742288900,"id_str":"471803285742288896","text":"RT @beyles66: cant wait for next year with
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285738094600,"id_str":"471803285738094592","text":"“@J_Mf_Gandee26: This picture 😆🍆👯🍻📱😩
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285767462900,"id_str":"471803285767462912","text":"El sábado es la reu y no voy..Segunda reun
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771649000,"id_str":"471803285771649024","text":"@Emabeltran8 Uuuuuu gatas?","source":"<a h
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771657200,"id_str":"471803285771657216","text":"En tu cara.","source":"<a href=\"http://tw
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737717760,"id_str":"471803285737717760","text":"@null 79","source":"<a href=\"http://twitt
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285741895700,"id_str":"471803285741895680","text":"「定期」フォローワーさん募集中。ー 神威","source":"<a h
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746106400,"id_str":"471803285746106368","text":"風呂にキュピトゥピキュピトゥピ入キュピ。","source":"<a hre
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285762883600,"id_str":"471803285762883585","text":"【定時】\nマルキュウマルマル\n\n学校の日は\n これから始まる
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733519360,"id_str":"471803285733519360","text":"Happy world burger day everyone!","source"
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737721860,"id_str":"471803285737721856","text":"9:00です☆今日のお天気何かな？居抜き店舗探しはhttp://
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733916700,"id_str":"471803285733916672","text":"RT @gagationer: Just ripped up my One Dire
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758701600,"id_str":"471803285758701568","text":"perro traidor","source":"web","truncated":
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758689300,"id_str":"471803285758689280","text":"Sumbar-Maroko Jajaki Kerjasama Perdagangan
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733531650,"id_str":"471803285733531649","text":"RT @RealTonyRocha: #twitter Twitter CEO Sa
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750304800,"id_str":"471803285750304768","text":"\"I love you more than Kanye West loves hi
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750300700,"id_str":"471803285750300672","text":"質問なんでもどうぞ。\nhttp://t.co/5FejhJtMFb","sc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758677000,"id_str":"471803285758676992","text":"RT @7sn37: ومو هامج شي !إلا ان في بعض عزيزتي
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733904400,"id_str":"471803285733904386","text":"Photo:  http://t.co/wy3XTCGw6u","source":"
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285754507260,"id_str":"471803285754507264","text":"2輪、国内メーカー 修理・新車販売してます。新車は店頭にも数
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737717760,"id_str":"471803285737717761","text":"\"@hanyacintadia: Aku selalu tersenyum dis
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742313500,"id_str":"471803285742313472","text":"RT @CanadiensMTL: 3. Ses adversaires sont
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746516000,"id_str":"471803285746515968","text":"RT @IntThings: The first step to changing
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771255800,"id_str":"471803285771255808","text":"RT @rogergzz: #HoyNoMePuedoLevantarConRoge
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771280400,"id_str":"471803285771280385","text":"RT @SoDamnReaI: this is so me http://t.co/
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285762887700,"id_str":"471803285762887680","text":"RT @RelatableQuote: I laughed way too much
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285763280900,"id_str":"471803285763280896","text":"RT @jordansaweirdo: finger speed RT @shayc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285754490900,"id_str":"471803285754490881","text":"@ChoiiSul ah kamu;(","source":"<a href=\"h
```

Normal text file   length : 347,731,425   lines : 110,001   Ln : 1   Col : 1   Pos : 1   Windows (CR LF)   UTF-8   INS



C:\Users\nachi\Desktop\Databases\Final_BD\tweets_550000.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
tweets_110000.txt
tweets_550000.txt

```
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746495500,"id_str":"471803285746495489","text":"There is no wealth but life. ~John Ruskin
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285738106900,"id_str":"471803285738106880","text":"Mucho la Plop esto, la Plop aquello, pero
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285767462900,"id_str":"471803285767462913","text":"motive. When a political idea finds its wa
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750681600,"id_str":"471803285750681600","text":"@im_2realbih bol!","source":"<a href=\"htt
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285759078400,"id_str":"471803285759078401","text":"A veces no entendemos por que, cuando, don
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742317600,"id_str":"471803285742317568","text":"jajajjaja hay no el vr todo por queres ver
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285763280900,"id_str":"471803285763280897","text":"@LariiNandoFT9 kkkkkkkkkkkkkk ai gente tc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742288900,"id_str":"471803285742288896","text":"RT @beyles66: cant wait for next year with
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285738094600,"id_str":"471803285738094592","text":"“@J_Mf_Gandee26: This picture 😆🍆👯🍻📱😩
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285767462900,"id_str":"471803285767462912","text":"El sábado es la reu y no voy..Segunda reun
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771649000,"id_str":"471803285771649024","text":"@Emabeltran8 Uuuuuu gatas?","source":"<a h
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771657200,"id_str":"471803285771657216","text":"En tu cara.","source":"<a href=\"http://tw
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737717760,"id_str":"471803285737717760","text":"@null 79","source":"<a href=\"http://twitt
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285741895700,"id_str":"471803285741895680","text":"「定期」フォローワーさん募集中。ー 神威","source":"<a h
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746106400,"id_str":"471803285746106368","text":"風呂にキュピトゥピキュピトゥピ入キュピ。","source":"<a hre
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285762883600,"id_str":"471803285762883585","text":"【定時】\nマルキュウマルマル\n\n学校の日は\n これから始まる
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733519360,"id_str":"471803285733519360","text":"Happy world burger day everyone!","source"
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737721860,"id_str":"471803285737721856","text":"9:00です☆今日のお天気何かな？居抜き店舗探しはhttp://
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733916700,"id_str":"471803285733916672","text":"RT @gagationer: Just ripped up my One Dire
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758701600,"id_str":"471803285758701568","text":"perro traidor","source":"web","truncated":
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758689300,"id_str":"471803285758689280","text":"Sumbar-Maroko Jajaki Kerjasama Perdagangan
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733531650,"id_str":"471803285733531649","text":"RT @RealTonyRocha: #twitter Twitter CEO Sa
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750304800,"id_str":"471803285750304768","text":"\"I love you more than Kanye West loves hi
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285750300700,"id_str":"471803285750300672","text":"質問なんでもどうぞ。\nhttp://t.co/5FejhJtMFb","sc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285758677000,"id_str":"471803285758676992","text":"RT @7sn37: ومو هامج شي !إلا ان في بعض عزيزتي
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285733904400,"id_str":"471803285733904386","text":"Photo:  http://t.co/wy3XTCGw6u","source":"
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285754507260,"id_str":"471803285754507264","text":"2輪、国内メーカー 修理・新車販売してます。新車は店頭にも数
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285737717760,"id_str":"471803285737717761","text":"\"@hanyacintadia: Aku selalu tersenyum dis
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285742313500,"id_str":"471803285742313472","text":"RT @CanadiensMTL: 3. Ses adversaires sont
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285746516000,"id_str":"471803285746515968","text":"RT @IntThings: The first step to changing
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771255800,"id_str":"471803285771255808","text":"RT @rogergzz: #HoyNoMePuedoLevantarConRoge
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285771280400,"id_str":"471803285771280385","text":"RT @SoDamnReaI: this is so me http://t.co/
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285762887700,"id_str":"471803285762887680","text":"RT @RelatableQuote: I laughed way too much
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285763280900,"id_str":"471803285763280896","text":"RT @jordansaweirdo: finger speed RT @shayc
{"created_at":"Thu May 29 00:00:43 +0000 2014","id":471803285754490900,"id_str":"471803285754490881","text":"@ChoiiSul ah kamu;(","source":"<a href=\"h
```

Normal text file   length : 1,737,380,787   lines : 550,001   Ln : 1   Col : 1   Pos : 1   Windows (CR LF)   UTF-8   INS

b. **Repeat what you did in part 1-a, but instead of saving tweets to the file, populate the 3-table schema that you previously created in SQLite. Be sure to execute commit and verify that the data has been successfully loaded. Report loaded row counts for each of the 3 tables.**

   NOTE: If your schema contains a foreign key in the Geo table or relies on TweetID as the primary key for the Geo table, you should change your schema. Geo entries should be identified based on the location they represent. There should not be any “blank” Geo entries such as (ID, None, None, None). The easiest way to create an ID is by combining lon_lat into a primary key.

CODE:

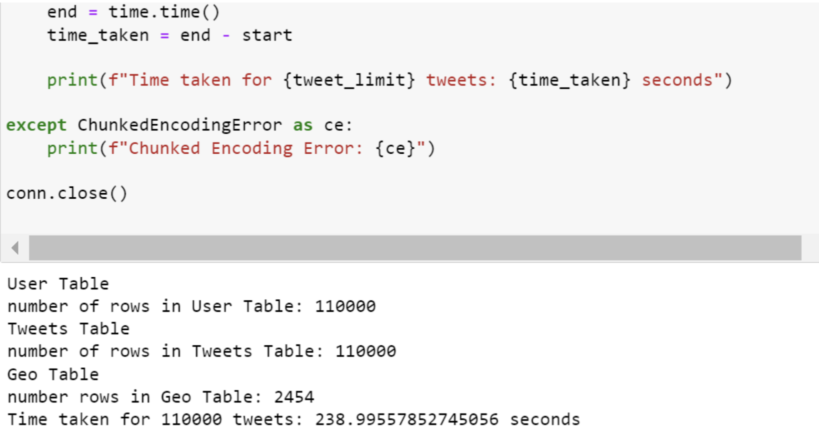
```
#1B
import sqlite3
import json
import time
import requests
from requests.exceptions import ChunkedEncodingError
```

```
url = "http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"
conn = sqlite3.connect('tweet_Final_550000_3.db')
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS Tweets")
cursor.execute("DROP TABLE IF EXISTS User")
cursor.execute("DROP TABLE IF EXISTS Geo")

cursor.execute('''CREATE TABLE IF NOT EXISTS User (
          user_id INTEGER PRIMARY KEY,
          id INTEGER,
          name VARCHAR(250),
          screen_name VARCHAR(250),
          description VARCHAR(300),
          friends_count INTEGER
        )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Geo (
          longitude REAL,
          latitude REAL,
          type VARCHAR(50),
          PRIMARY KEY (longitude, latitude)
        )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Tweets (
          created_at VARCHAR(50),
          id_str INTEGER,
          text VARCHAR(300),
          source VARCHAR(100),
          in_reply_to_user_id INTEGER,
          in_reply_to_screen_name VARCHAR(150),
          in_reply_to_status_id INTEGER,
          retweet_count INTEGER,
          contributors VARCHAR(100),
          user_id INTEGER,
          longitude REAL,
          latitude REAL,
          FOREIGN KEY (user_id) REFERENCES User (user_id),
          FOREIGN KEY (longitude, latitude) REFERENCES Geo (longitude, latitude)
        )''')

tweet_limit = 550000
tweet_counter = 0

start = time.time()

try:
    response = requests.get(url, stream=True)
    lines = response.iter_lines()

    error_tweets = []
    geo_dict = {}

    for line in lines:
        if tweet_counter >= tweet_limit:
            break

        try:
            decoded_line = line.decode('utf-8')
            tweet_dict = json.loads(decoded_line)

            user_id = tweet_dict['user']['id']
            name = tweet_dict['user']['name']
            screen_name = tweet_dict['user']['screen_name']
            description = tweet_dict['user']['description']
            friends_count = tweet_dict['user']['friends_count']

            cursor.execute('''INSERT INTO User (id, name, screen_name, description, friends_count)
                    VALUES (?, ?, ?, ?, ?)''',
                           (user_id, name, screen_name, description, friends_count))
            user_row_id = cursor.lastrowid

            created_at = tweet_dict['created_at']
            id_str = tweet_dict['id_str']
            text = tweet_dict['text']
            source = tweet_dict['source']
            in_reply_to_user_id = tweet_dict['in_reply_to_user_id']
            in_reply_to_screen_name = tweet_dict['in_reply_to_screen_name']
            in_reply_to_status_id = tweet_dict['in_reply_to_status_id']
            retweet_count = tweet_dict['retweet_count']
            contributors = tweet_dict['contributors']

            geo_data = tweet_dict.get('geo')
            if geo_data is not None:
                geo_type = geo_data['type']
                longitude, latitude = geo_data['coordinates']
            else:
                geo_type = None
                longitude = None
                latitude = None

            if longitude is not None and latitude is not None:
                geo_key = f'{longitude}_{latitude}'
                if geo_key in geo_dict:
                    geo_id = geo_dict[geo_key]
                else:
                    cursor.execute('''INSERT INTO Geo (longitude, latitude, type)
                                   VALUES (?, ?, ?)''',
                                   (longitude, latitude, geo_type))
                    geo_id = (longitude, latitude)
                    geo_dict[geo_key] = geo_id
            else:
                geo_id = None

            cursor.execute('''INSERT INTO Tweets (created_at, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_name, in_reply_to_status_id, retweet_count, contributors, user_id, longitude, latitude)
                            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)''',
                           (created_at, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_name, in_reply_to_status_id, retweet_count, contributors, user_id, longitude, latitude))
            conn.commit()

            tweet_counter += 1

        except Exception as e:
            error_tweets.append(line)
            print(f"Error: {e}")

    print("User Table")
    cursor.execute('SELECT COUNT(*) FROM User')
    user_count = cursor.fetchone()[0]
    print("number of rows in User Table:", user_count)

    print("Tweets Table")
    cursor.execute('SELECT COUNT(*) FROM Tweets')
    tweet_count = cursor.fetchone()[0]
    print("number of rows in Tweets Table:", tweet_count)

    print("Geo Table")
    cursor.execute('SELECT COUNT(*) FROM Geo')
    geo_count = cursor.fetchone()[0]
    print("number rows in Geo Table:", geo_count)

    end = time.time()
    time_taken = end - start

    print(f"Time taken for {tweet_limit} tweets: {time_taken} seconds")

except ChunkedEncodingError as ce:
    print(f"Chunked Encoding Error: {ce}")
```

```
conn.close()
```

O/P:

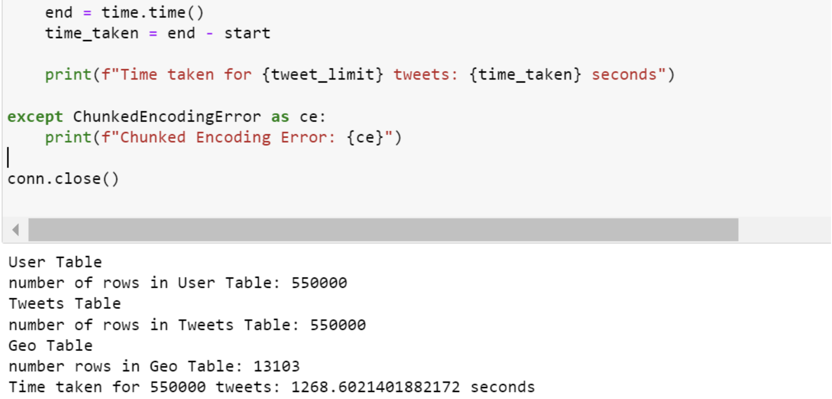
FOR 110000 TWEETS:



```
    end = time.time()
    time_taken = end - start

    print(f"Time taken for {tweet_limit} tweets: {time_taken} seconds")

except ChunkedEncodingError as ce:
    print(f"Chunked Encoding Error: {ce}")

conn.close()
```

```
User Table
number of rows in User Table: 110000
Tweets Table
number of rows in Tweets Table: 110000
Geo Table
number rows in Geo Table: 2454
Time taken for 110000 tweets: 238.99557852745056 seconds
```

FOR 550000 TWEETS:



```
    end = time.time()
    time_taken = end - start

    print(f"Time taken for {tweet_limit} tweets: {time_taken} seconds")

except ChunkedEncodingError as ce:
    print(f"Chunked Encoding Error: {ce}")

conn.close()
```

```
User Table
number of rows in User Table: 550000
Tweets Table
number of rows in Tweets Table: 550000
Geo Table
number rows in Geo Table: 13103
Time taken for 550000 tweets: 1268.6021401882172 seconds
```

c. **Use your locally saved tweet file to repeat the database population step from part-c. That is, load the tweets into the 3-table database using your saved file with tweets. This is the same code as in 1-b, but reading tweets from your file, not from the web.**

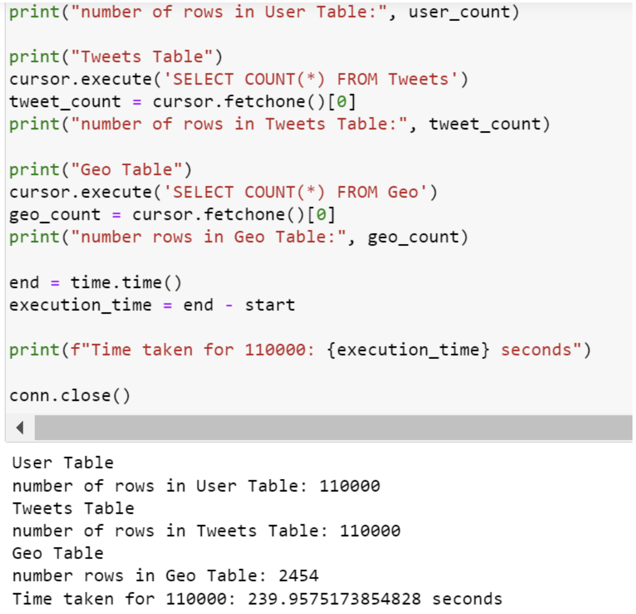
CODE:

```
#1C
import sqlite3
import json
import time

file_path = "C:/Users/nachi/Desktop/Databases/Final_BD/tweets_550000.txt" # Update with the actual path to your saved tweet file
conn = sqlite3.connect('tweet_1C_550000.db')
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS Tweets")
cursor.execute("DROP TABLE IF EXISTS User")
cursor.execute("DROP TABLE IF EXISTS Geo")

cursor.execute("'CREATE TABLE IF NOT EXISTS User (
            user_id INTEGER PRIMARY KEY,
            id INTEGER,
            name VARCHAR(250),
            screen_name VARCHAR(250),
            description VARCHAR(300),
            friends_count INTEGER
```

```
                )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Geo (
                    longitude REAL,
                    latitude REAL,
                    type VARCHAR(50),
                    PRIMARY KEY (longitude, latitude)
                )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Tweets (
                    created_at VARCHAR(50),
                    id_str INTEGER,
                    text VARCHAR(300),
                    source VARCHAR(100),
                    in_reply_to_user_id INTEGER,
                    in_reply_to_screen_name VARCHAR(150),
                    in_reply_to_status_id INTEGER,
                    retweet_count INTEGER,
                    contributors VARCHAR(100),
                    user_id INTEGER,
                    longitude REAL,
                    latitude REAL,
                    FOREIGN KEY (user_id) REFERENCES User (user_id),
                    FOREIGN KEY (longitude, latitude) REFERENCES Geo (longitude, latitude)
                )''')

start = time.time()

with open(file_path, "r", encoding="utf-8") as file:
    error_tweets = []
    geo_dict = {}

    for line in file:
        try:
            tweet_dict = json.loads(line)

            user_id = tweet_dict['user']['id']
            name = tweet_dict['user']['name']
            screen_name = tweet_dict['user']['screen_name']
            description = tweet_dict['user']['description']
            friends_count = tweet_dict['user']['friends_count']

            cursor.execute('''INSERT INTO User (id, name, screen_name, description, friends_count)
                            VALUES (?, ?, ?, ?, ?)''',
                           (user_id, name, screen_name, description, friends_count))
            user_row_id = cursor.lastrowid

            created_at = tweet_dict['created_at']
            id_str = tweet_dict['id_str']
            text = tweet_dict['text']
            source = tweet_dict['source']
            in_reply_to_user_id = tweet_dict['in_reply_to_user_id']
            in_reply_to_screen_name = tweet_dict['in_reply_to_screen_name']
            in_reply_to_status_id = tweet_dict['in_reply_to_status_id']
            retweet_count = tweet_dict['retweet_count']
            contributors = tweet_dict['contributors']

            geo_data = tweet_dict.get('geo')
            if geo_data is not None:
                geo_type = geo_data['type']
                longitude, latitude = geo_data['coordinates']
            else:
                geo_type = None
                longitude = None
                latitude = None

            if longitude is not None and latitude is not None:
                geo_key = f'{longitude}_{latitude}'
                if geo_key in geo_dict:
                    geo_id = geo_dict[geo_key]
                else:
                    cursor.execute('''INSERT INTO Geo (longitude, latitude, type)
                                VALUES (?, ?, ?)''',
                                (longitude, latitude, geo_type))
                    geo_id = (longitude, latitude)
                    geo_dict[geo_key] = geo_id
            else:
                geo_id = None

            cursor.execute('''INSERT INTO Tweets (created_at, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_name,
in_reply_to_status_id, retweet_count, contributors, user_id, longitude, latitude)
                            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)''',
                           (created_at, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_name, in_reply_to_status_id, retweet_count,
contributors, user_id, longitude, latitude))
            conn.commit()

        except Exception as e:
            error_tweets.append(line)
            print(f"Error: {e}")

print("User Table")
cursor.execute('SELECT COUNT(*) FROM User')
user_count = cursor.fetchone()[0]
print("number of rows in User Table:", user_count)

print("Tweets Table")
cursor.execute('SELECT COUNT(*) FROM Tweets')
tweet_count = cursor.fetchone()[0]
print("number of rows in Tweets Table:", tweet_count)

print("Geo Table")
cursor.execute('SELECT COUNT(*) FROM Geo')
geo_count = cursor.fetchone()[0]
print("number rows in Geo Table:", geo_count)

end = time.time()
execution_time = end - start

print(f"Time taken for 550000: {execution_time} seconds")

conn.close()
```

O/P:

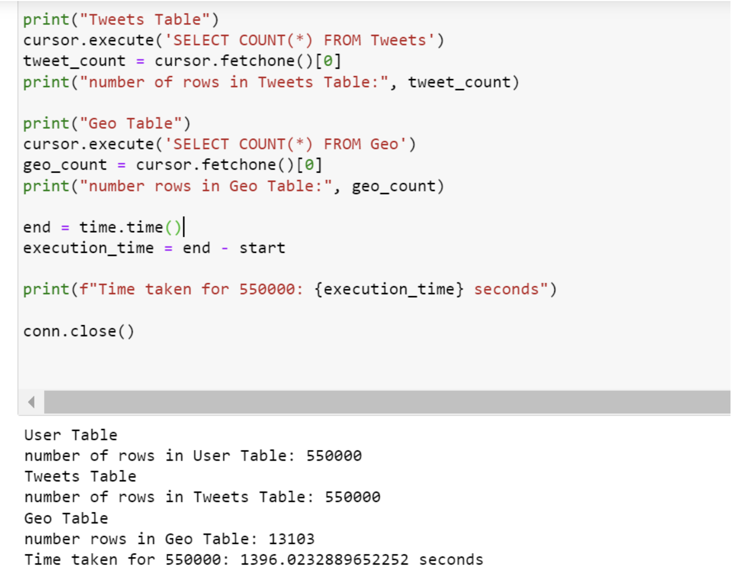
FOR 110000 TWEETS:



```
print("number of rows in User Table:", user_count)

print("Tweets Table")
cursor.execute('SELECT COUNT(*) FROM Tweets')
tweet_count = cursor.fetchone()[0]
print("number of rows in Tweets Table:", tweet_count)

print("Geo Table")
cursor.execute('SELECT COUNT(*) FROM Geo')
geo_count = cursor.fetchone()[0]
print("number rows in Geo Table:", geo_count)

end = time.time()
execution_time = end - start

print(f"Time taken for 110000: {execution_time} seconds")

conn.close()
```

```
User Table
number of rows in User Table: 110000
Tweets Table
number of rows in Tweets Table: 110000
Geo Table
number rows in Geo Table: 2454
Time taken for 110000: 239.9575173854828 seconds
```

FOR 550000 TWEETS:



```
print("Tweets Table")
cursor.execute('SELECT COUNT(*) FROM Tweets')
tweet_count = cursor.fetchone()[0]
print("number of rows in Tweets Table:", tweet_count)

print("Geo Table")
cursor.execute('SELECT COUNT(*) FROM Geo')
geo_count = cursor.fetchone()[0]
print("number rows in Geo Table:", geo_count)

end = time.time()
execution_time = end - start

print(f"Time taken for 550000: {execution_time} seconds")

conn.close()
```

```
User Table
number of rows in User Table: 550000
Tweets Table
number of rows in Tweets Table: 550000
Geo Table
number rows in Geo Table: 13103
Time taken for 550000: 1396.0232889652252 seconds
```

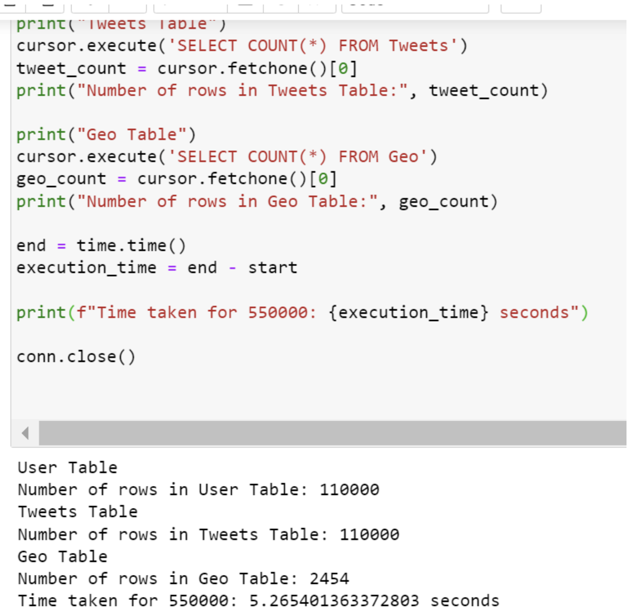
d. **Repeat the same step with a batching size of 2,000 (i.e. by inserting 2,000 rows at a time with executemany instead of doing individual inserts). Since many of the tweets are missing a Geo location, its fine for the batches of Geo inserts to be smaller than 2,000.**

CODE:

```
#1D 110000 TWEETS
import sqlite3
import json
import time

file_path = "C:/Users/nachi/Desktop/Databases/Final_BD/tweets_110000.txt"  # Update with the actual path to your saved tweet file
conn = sqlite3.connect('tweet_1D_110000.db')
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS Tweets")
cursor.execute("DROP TABLE IF EXISTS User")
cursor.execute("DROP TABLE IF EXISTS Geo")

cursor.execute('''CREATE TABLE IF NOT EXISTS User (
            user_id INTEGER PRIMARY KEY,
            id INTEGER,
            name VARCHAR(250),
            screen_name VARCHAR(250),
            description VARCHAR(300),
            friends_count INTEGER
        )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Geo (
            longitude REAL,
            latitude REAL,
            type VARCHAR(50),
            PRIMARY KEY (longitude, latitude)
        )''')

cursor.execute('''CREATE TABLE IF NOT EXISTS Tweets (
                    created_at VARCHAR(50),
                    id_str INTEGER,
                    text VARCHAR(300),
                    source VARCHAR(100),
                    in_reply_to_user_id INTEGER,
                    in_reply_to_screen_name VARCHAR(150),
                    in_reply_to_status_id INTEGER,
                    retweet_count INTEGER,
                    contributors VARCHAR(100),
                    user_id INTEGER,
                    longitude REAL,
                    latitude REAL,
                    FOREIGN KEY (user_id) REFERENCES User (user_id),
                    FOREIGN KEY (longitude, latitude) REFERENCES Geo (longitude, latitude)
                )''')

start = time.time()

with open(file_path, "r", encoding="utf-8") as file:
    error_tweets = []
    geo_dict = {}
    set_batch = 2000
    tweet_batch = []
    geo_batch = []

    for line in file:
        try:
            tweet_dict = json.loads(line)

            user_id = tweet_dict['user']['id']
            name = tweet_dict['user']['name']
            screen_name = tweet_dict['user']['screen_name']
            description = tweet_dict['user']['description']
            friends_count = tweet_dict['user']['friends_count']

            cursor.execute('''INSERT INTO User (id, name, screen_name, description, friends_count)
                            VALUES (?, ?, ?, ?, ?)''',
                           (user_id, name, screen_name, description, friends_count))
            user_row_id = cursor.lastrowid

            created_at = tweet_dict['created_at']
            id_str = tweet_dict['id_str']
            text = tweet_dict['text']
            source = tweet_dict['source']
            in_reply_to_user_id = tweet_dict['in_reply_to_user_id']
            in_reply_to_screen_name = tweet_dict['in_reply_to_screen_name']
            in_reply_to_status_id = tweet_dict['in_reply_to_status_id']
            retweet_count = tweet_dict['retweet_count']
            contributors = tweet_dict['contributors']

            geo_data = tweet_dict.get('geo')
            if geo_data is not None:
                geo_type = geo_data['type']
                longitude, latitude = geo_data['coordinates']
                geo_key = (longitude, latitude)

                if geo_key not in geo_dict:
                    geo_dict[geo_key] = geo_type
                    geo_batch.append((longitude, latitude, geo_type))
            else:
                geo_type = None
                longitude = None
                latitude = None

            tweet_batch.append((created_at, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_name,
                                in_reply_to_status_id, retweet_count, contributors, user_id, longitude, latitude))

            if len(tweet_batch) >= set_batch:
                cursor.executemany('''INSERT INTO Geo (longitude, latitude, type)
                                    VALUES (?, ?, ?)''', geo_batch)
                    conn.commit()
                    geo_batch.clear()
                    cursor.executemany('''INSERT INTO Tweets (created_at, id_str, text, source, in_reply_to_user_id,
                                                            in_reply_to_screen_name, in_reply_to_status_id, retweet_count,
                                                            contributors, user_id, longitude, latitude)
                                        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)''', tweet_batch)
                    conn.commit()
                    tweet_batch.clear()

            except Exception as e:
                error_tweets.append(line)
                print(f"Error: {e}")
        if tweet_batch:
            cursor.executemany('''INSERT INTO Tweets (created_at, id_str, text, source, in_reply_to_user_id,
                                                    in_reply_to_screen_name, in_reply_to_status_id, retweet_count,
                                                    contributors, user_id, longitude, latitude)
                                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)''', tweet_batch)
            conn.commit()
            tweet_batch.clear()
        if geo_batch:
            cursor.executemany('''INSERT INTO Geo (longitude, latitude, type)
                                VALUES (?, ?, ?)''', geo_batch)
            conn.commit()
            geo_batch.clear()

    print("User Table")
    cursor.execute('SELECT COUNT(*) FROM User')
    user_count = cursor.fetchone()[0]
    print("Number of rows in User Table:", user_count)
    print("Tweets Table")
    cursor.execute('SELECT COUNT(*) FROM Tweets')
    tweet_count = cursor.fetchone()[0]
    print("Number of rows in Tweets Table:", tweet_count)
    print("Geo Table")
    cursor.execute('SELECT COUNT(*) FROM Geo')
    geo_count = cursor.fetchone()[0]
    print("Number of rows in Geo Table:", geo_count)

    end = time.time()
    execution_time = end - start

    print(f"Time taken for 550000: {execution_time} seconds")

conn.close()
```

O/P:

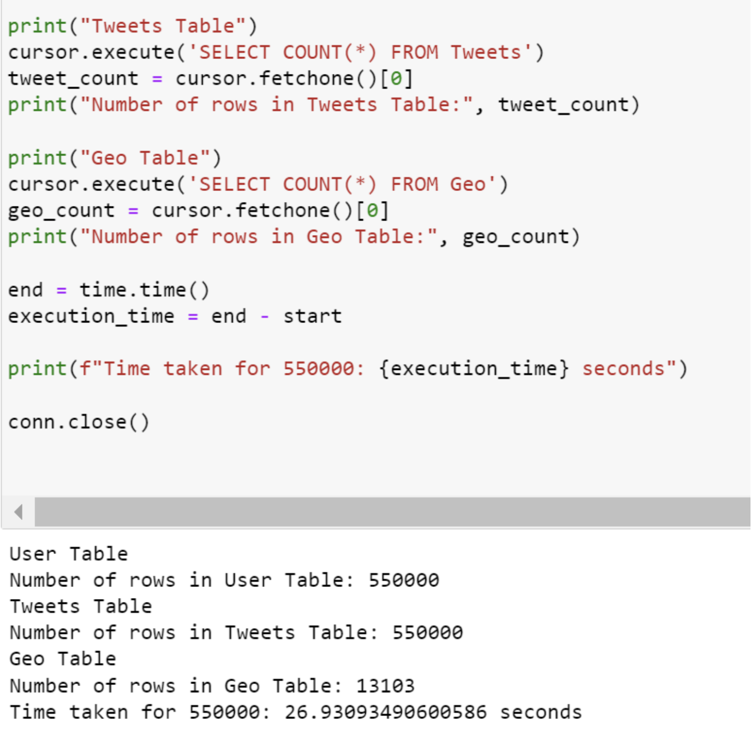
FOR 110000 TWEETS:



```
print("Tweets Table")
cursor.execute('SELECT COUNT(*) FROM Tweets')
tweet_count = cursor.fetchone()[0]
print("Number of rows in Tweets Table:", tweet_count)

print("Geo Table")
cursor.execute('SELECT COUNT(*) FROM Geo')
geo_count = cursor.fetchone()[0]
print("Number of rows in Geo Table:", geo_count)

end = time.time()
execution_time = end - start

print(f"Time taken for 550000: {execution_time} seconds")

conn.close()
```

```
User Table
Number of rows in User Table: 110000
Tweets Table
Number of rows in Tweets Table: 110000
Geo Table
Number of rows in Geo Table: 2454
Time taken for 550000: 5.265401363372803 seconds
```

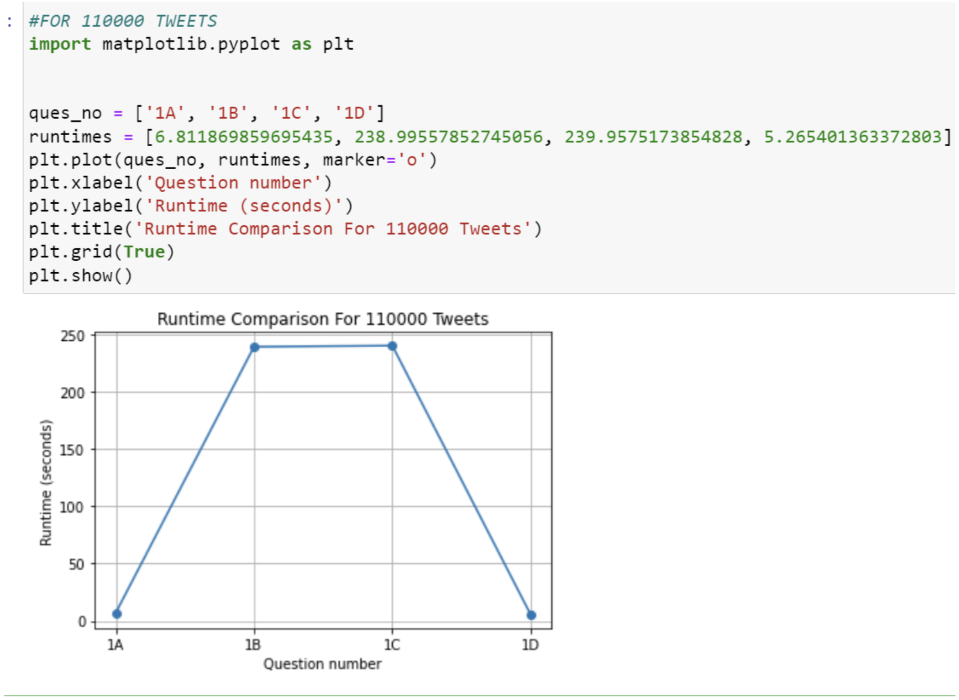
FOR 550000 TWEETS:



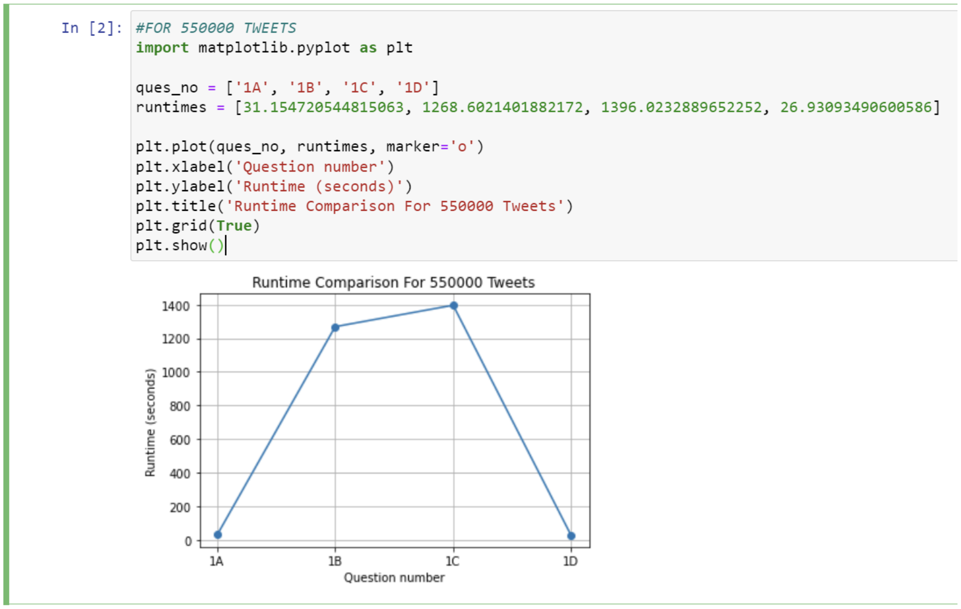
```
print("Tweets Table")
cursor.execute('SELECT COUNT(*) FROM Tweets')
tweet_count = cursor.fetchone()[0]
print("Number of rows in Tweets Table:", tweet_count)

print("Geo Table")
cursor.execute('SELECT COUNT(*) FROM Geo')
geo_count = cursor.fetchone()[0]
print("Number of rows in Geo Table:", geo_count)

end = time.time()
execution_time = end - start

print(f"Time taken for 550000: {execution_time} seconds")

conn.close()
```

```
User Table
Number of rows in User Table: 550000
Tweets Table
Number of rows in Tweets Table: 550000
Geo Table
Number of rows in Geo Table: 13103
Time taken for 550000: 26.93093490600586 seconds
```

e. **Plot the resulting runtimes (# of tweets versus runtimes) using matplotlib for 1-a, 1-b, 1-c, and 1-d. How does the runtime compare?**



```
#FOR 110000 TWEETS
import matplotlib.pyplot as plt


ques_no = ['1A', '1B', '1C', '1D']
runtimes = [6.811869859695435, 238.99557852745056, 239.9575173854828, 5.265401363372803]
plt.plot(ques_no, runtimes, marker='o')
plt.xlabel('Question number')
plt.ylabel('Runtime (seconds)')
plt.title('Runtime Comparison For 110000 Tweets')
plt.grid(True)
plt.show()
```

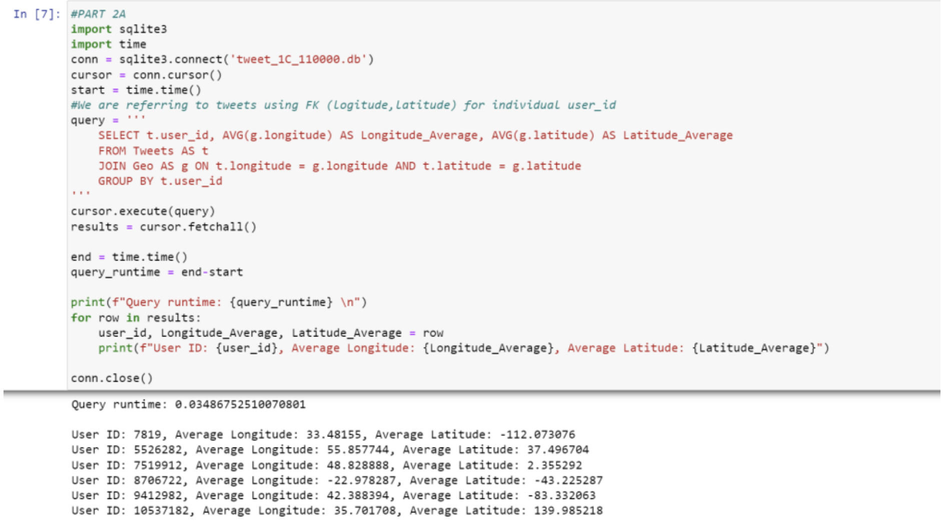


```
#FOR 550000 TWEETS
import matplotlib.pyplot as plt

ques_no = ['1A', '1B', '1C', '1D']
runtimes = [31.154720544815063, 1268.6021401882172, 1396.0232889652252, 26.93093490600586]

plt.plot(ques_no, runtimes, marker='o')
plt.xlabel('Question number')
plt.ylabel('Runtime (seconds)')
plt.title('Runtime Comparison For 550000 Tweets')
plt.grid(True)
plt.show()
```

From the above graphs we can observe that from 1B and 1C (Since the files are large) that data being loaded to database from the link and the files took the most runtime. Whereas when the data was split into chucks it took very little time compared to 1B and 1C.

Part 2

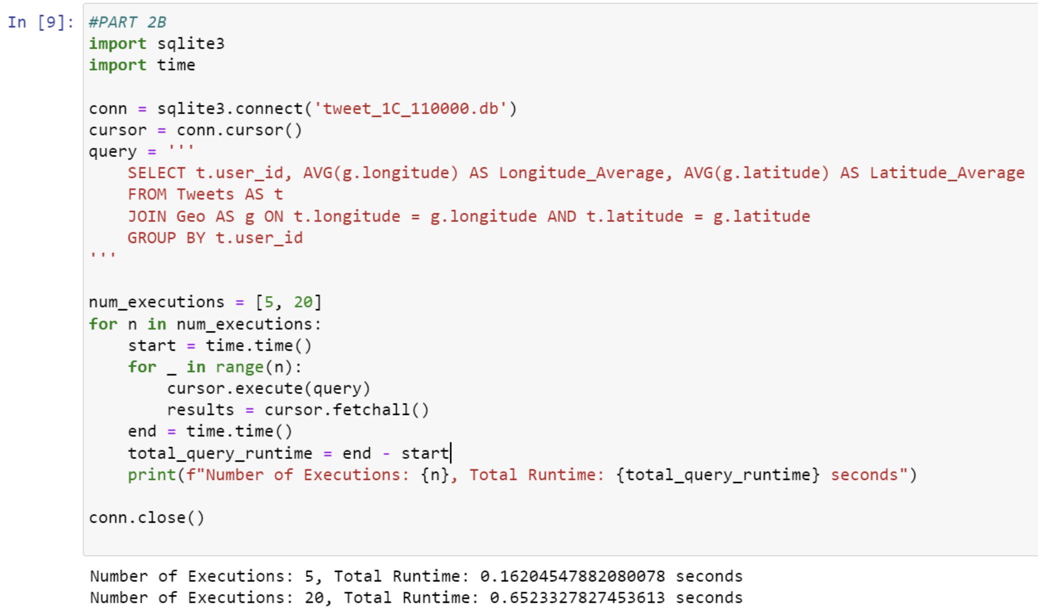
a. **Write and execute a SQL query to find the average longitude and latitude value for each user ID. This query does not need the User table because User ID is a foreign key in the Tweet table. E.g., something like *SELECT UserID, MIN(longitude), MAX(latitude) FROM Tweet, Geo WHERE Tweet.GeoFK = Geo.GeoID GROUP BY UserID;***



```
In [7]: #PART 2A
        import sqlite3
        import time
        conn = sqlite3.connect('tweet_1C_110000.db')
        cursor = conn.cursor()
        start = time.time()
        #We are referring to tweets using FK (logitude,latitude) for individual user_id
        query = '''
            SELECT t.user_id, AVG(g.longitude) AS Longitude_Average, AVG(g.latitude) AS Latitude_Average
            FROM Tweets AS t
            JOIN Geo AS g ON t.longitude = g.longitude AND t.latitude = g.latitude
            GROUP BY t.user_id
        '''
        cursor.execute(query)
        results = cursor.fetchall()

        end = time.time()
        query_runtime = end-start

        print(f"Query runtime: {query_runtime} \n")
        for row in results:
            user_id, Longitude_Average, Latitude_Average = row
            print(f"User ID: {user_id}, Average Longitude: {Longitude_Average}, Average Latitude: {Latitude_Average}")

        conn.close()
```

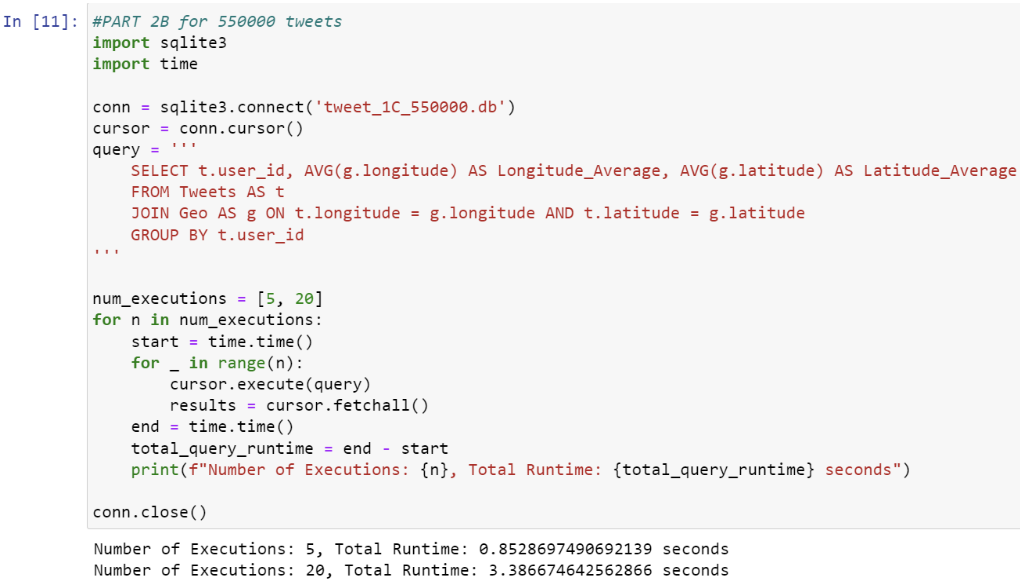
```
Query runtime: 0.03486752510070801

User ID: 7819, Average Longitude: 33.48155, Average Latitude: -112.073076
User ID: 5526282, Average Longitude: 55.857744, Average Latitude: 37.496704
User ID: 7519912, Average Longitude: 48.828888, Average Latitude: 2.355292
User ID: 8706722, Average Longitude: -22.978287, Average Latitude: -43.225287
User ID: 9412982, Average Longitude: 42.388394, Average Latitude: -83.332063
User ID: 10537182, Average Longitude: 35.701708, Average Latitude: 139.985218
```

b. **Re-execute the SQL query in part 2-a 5 times and 20 times and measure the total runtime (just re-run the same exact query multiple times using a for-loop, it is as simple as it looks). Does the runtime scale linearly? (i.e., does it take 5X and 20X as much time?)**



```
In [9]: #PART 2B
        import sqlite3
        import time

        conn = sqlite3.connect('tweet_1C_110000.db')
        cursor = conn.cursor()
        query = '''
            SELECT t.user_id, AVG(g.longitude) AS Longitude_Average, AVG(g.latitude) AS Latitude_Average
            FROM Tweets AS t
            JOIN Geo AS g ON t.longitude = g.longitude AND t.latitude = g.latitude
            GROUP BY t.user_id
        '''

        num_executions = [5, 20]
        for n in num_executions:
            start = time.time()
            for _ in range(n):
                cursor.execute(query)
                results = cursor.fetchall()
            end = time.time()
            total_query_runtime = end - start
            print(f"Number of Executions: {n}, Total Runtime: {total_query_runtime} seconds")

        conn.close()
```

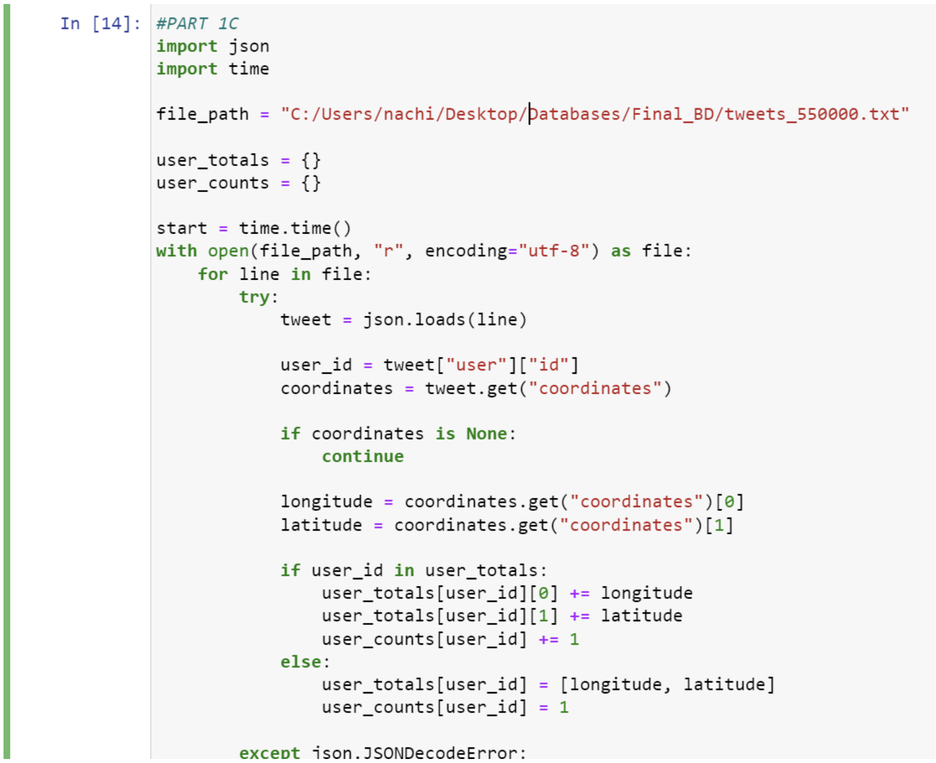
```
Number of Executions: 5, Total Runtime: 0.16204547882080078 seconds
Number of Executions: 20, Total Runtime: 0.6523327827453613 seconds
```



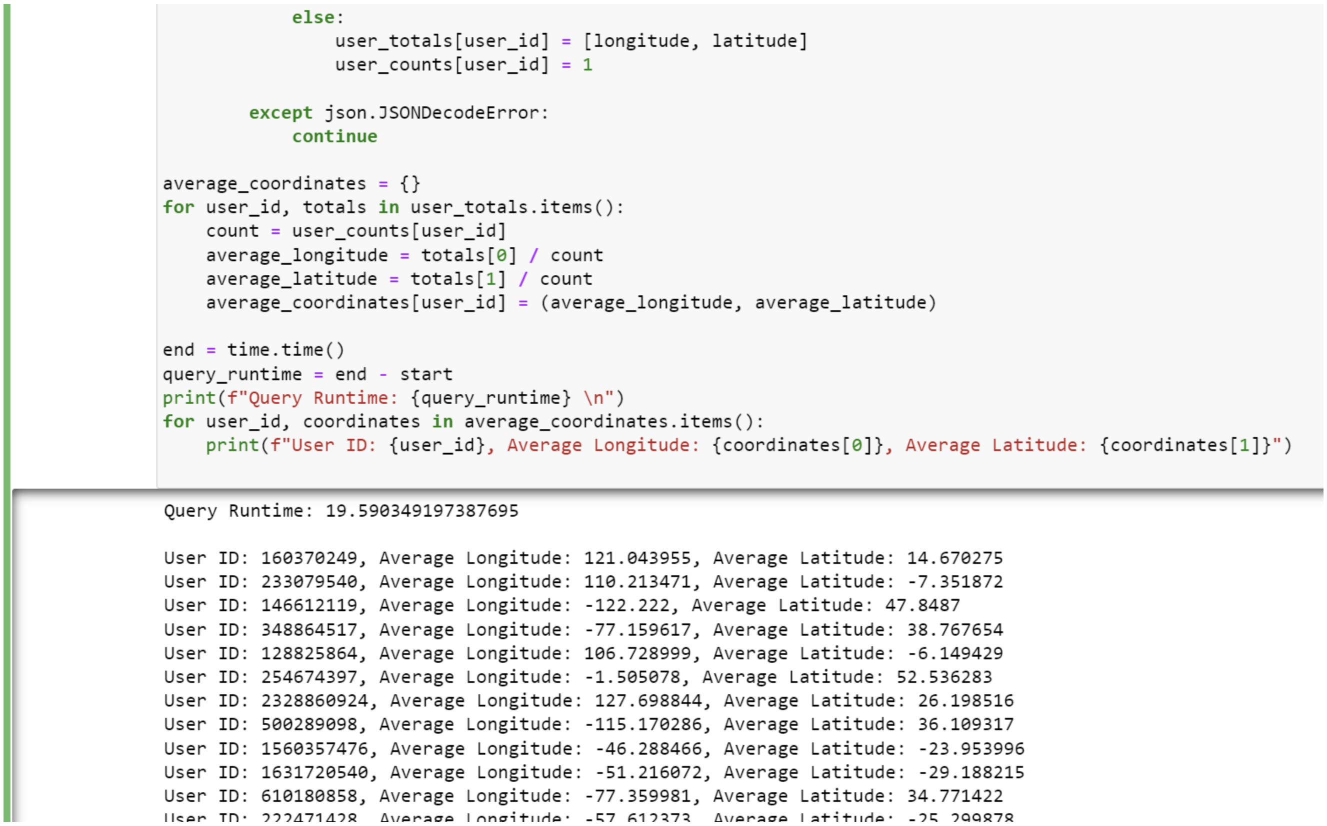
```
In [11]: #PART 2B for 550000 tweets
         import sqlite3
         import time

         conn = sqlite3.connect('tweet_1C_550000.db')
         cursor = conn.cursor()
         query = '''
             SELECT t.user_id, AVG(g.longitude) AS Longitude_Average, AVG(g.latitude) AS Latitude_Average
             FROM Tweets AS t
             JOIN Geo AS g ON t.longitude = g.longitude AND t.latitude = g.latitude
             GROUP BY t.user_id
         '''

         num_executions = [5, 20]
         for n in num_executions:
             start = time.time()
             for _ in range(n):
                 cursor.execute(query)
                 results = cursor.fetchall()
             end = time.time()
             total_query_runtime = end - start
             print(f"Number of Executions: {n}, Total Runtime: {total_query_runtime} seconds")

         conn.close()

         Number of Executions: 5, Total Runtime: 0.8528697490692139 seconds
         Number of Executions: 20, Total Runtime: 3.386674642562866 seconds
```

Answer 2B: For 110000 tweets we can see the runtime scales linearly whereas for 550000 tweets the runtime the ratio is approximately 3.99 (Expected ratio 20/5 = 4), which is very close to the expected value. Therefore, based on these results, we can conclude that the runtime scales approximately linearly with the number of executions.

c. **Write the equivalent of the 2-a query in python (without using SQL) by reading it from the file with 550,000 tweets.**



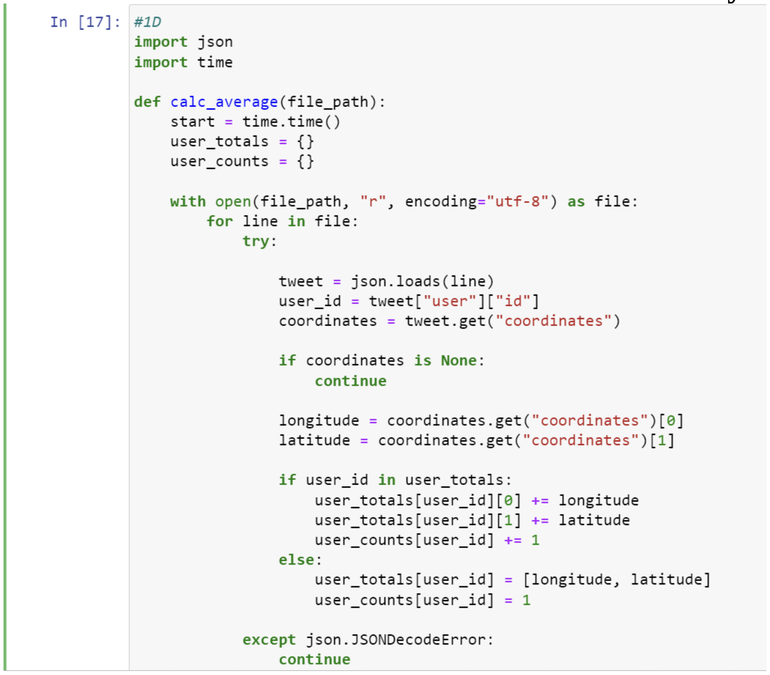
```
In [14]: #PART 1C
         import json
         import time

         file_path = "C:/Users/nachi/Desktop/Databases/Final_BD/tweets_550000.txt"

         user_totals = {}
         user_counts = {}

         start = time.time()
         with open(file_path, "r", encoding="utf-8") as file:
             for line in file:
                 try:
                     tweet = json.loads(line)

                     user_id = tweet["user"]["id"]
                     coordinates = tweet.get("coordinates")

                     if coordinates is None:
                         continue

                     longitude = coordinates.get("coordinates")[0]
                     latitude = coordinates.get("coordinates")[1]

                     if user_id in user_totals:
                         user_totals[user_id][0] += longitude
                         user_totals[user_id][1] += latitude
                         user_counts[user_id] += 1
                     else:
                         user_totals[user_id] = [longitude, latitude]
                         user_counts[user_id] = 1

                 except json.JSONDecodeError:
```



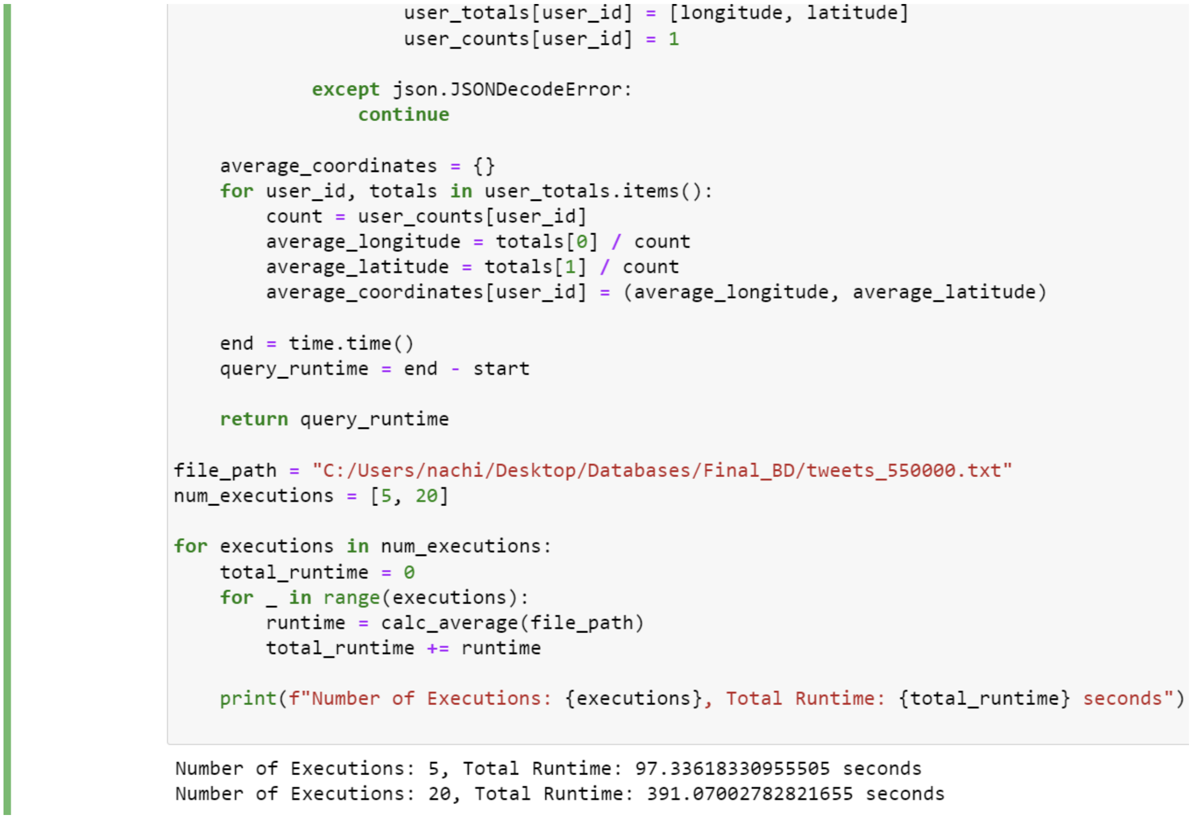
```
            else:
                user_totals[user_id] = [longitude, latitude]
                user_counts[user_id] = 1

        except json.JSONDecodeError:
            continue

average_coordinates = {}
for user_id, totals in user_totals.items():
    count = user_counts[user_id]
    average_longitude = totals[0] / count
    average_latitude = totals[1] / count
    average_coordinates[user_id] = (average_longitude, average_latitude)

end = time.time()
query_runtime = end - start
print(f"Query Runtime: {query_runtime} \n")
for user_id, coordinates in average_coordinates.items():
    print(f"User ID: {user_id}, Average Longitude: {coordinates[0]}, Average Latitude: {coordinates[1]}")
```

```
Query Runtime: 19.590349197387695

User ID: 160370249, Average Longitude: 121.043955, Average Latitude: 14.670275
User ID: 233079540, Average Longitude: 110.213471, Average Latitude: -7.351872
User ID: 146612119, Average Longitude: -122.222, Average Latitude: 47.8487
User ID: 348864517, Average Longitude: -77.159617, Average Latitude: 38.767654
User ID: 128825864, Average Longitude: 106.728999, Average Latitude: -6.149429
User ID: 254674397, Average Longitude: -1.505078, Average Latitude: 52.536283
User ID: 2328860924, Average Longitude: 127.698844, Average Latitude: 26.198516
User ID: 500289098, Average Longitude: -115.170286, Average Latitude: 36.109317
User ID: 1560357476, Average Longitude: -46.288466, Average Latitude: -23.953996
User ID: 1631720540, Average Longitude: -51.216072, Average Latitude: -29.188215
User ID: 610180858, Average Longitude: -77.359981, Average Latitude: 34.771422
User ID: 222471428, Average Longitude: -57.612373, Average Latitude: -25.299878
```

d. Re-execute the query in part 2-c 5 times and 20 times and measure the total runtime. Does the runtime scale linearly?



```
In [17]: #1D
         import json
         import time

         def calc_average(file_path):
             start = time.time()
             user_totals = {}
             user_counts = {}

             with open(file_path, "r", encoding="utf-8") as file:
                 for line in file:
                     try:

                         tweet = json.loads(line)
                         user_id = tweet["user"]["id"]
                         coordinates = tweet.get("coordinates")

                         if coordinates is None:
                             continue

                         longitude = coordinates.get("coordinates")[0]
                         latitude = coordinates.get("coordinates")[1]

                         if user_id in user_totals:
                             user_totals[user_id][0] += longitude
                             user_totals[user_id][1] += latitude
                             user_counts[user_id] += 1
                         else:
                             user_totals[user_id] = [longitude, latitude]
                             user_counts[user_id] = 1

                     except json.JSONDecodeError:
                         continue
```

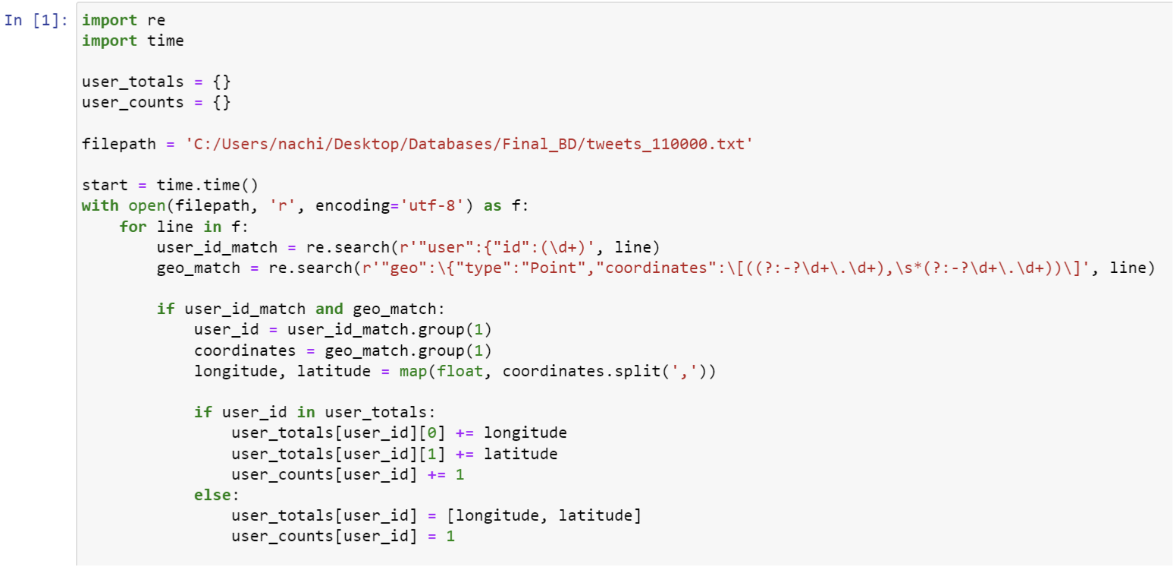


```
                    user_totals[user_id] = [longitude, latitude]
                    user_counts[user_id] = 1

            except json.JSONDecodeError:
                continue

    average_coordinates = {}
    for user_id, totals in user_totals.items():
        count = user_counts[user_id]
        average_longitude = totals[0] / count
        average_latitude = totals[1] / count
        average_coordinates[user_id] = (average_longitude, average_latitude)

    end = time.time()
    query_runtime = end - start

    return query_runtime

file_path = "C:/Users/nachi/Desktop/Databases/Final_BD/tweets_550000.txt"
num_executions = [5, 20]

for executions in num_executions:
    total_runtime = 0
    for _ in range(executions):
        runtime = calc_average(file_path)
        total_runtime += runtime

    print(f"Number of Executions: {executions}, Total Runtime: {total_runtime} seconds")
```

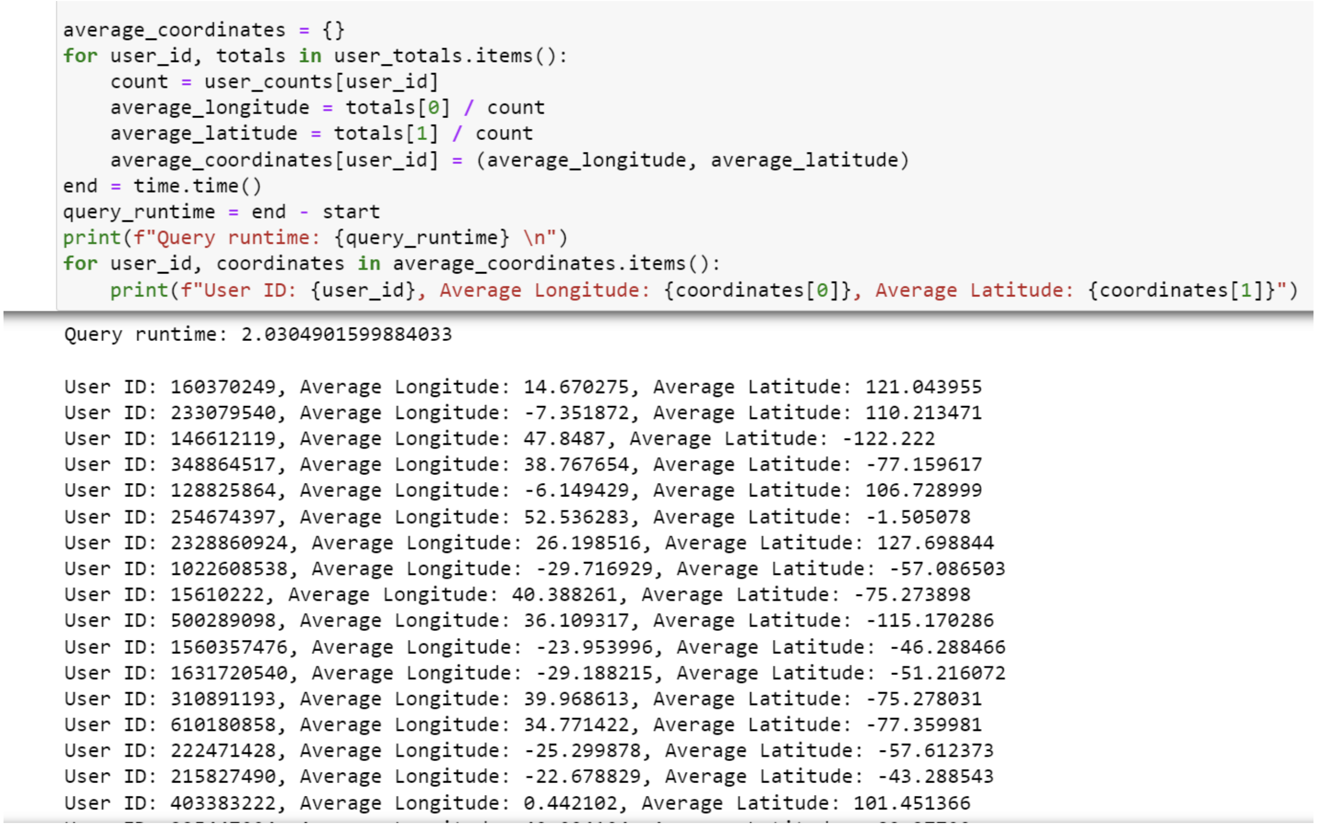
```
Number of Executions: 5, Total Runtime: 97.33618330955505 seconds
Number of Executions: 20, Total Runtime: 391.07002782821655 seconds
```

Answer 2D: The ratio is approximately 4.02, which is close to the expected value. Therefore, we can say that the runtime scales approximately linearly with the number of executions.

e. **Write the equivalent of the 2-a query in python by using regular expressions instead of json.loads(). Do not use json.loads() here. Note that you only need to find userid and geo location (if any) for each tweet, you don't need to parse the whole thing.**



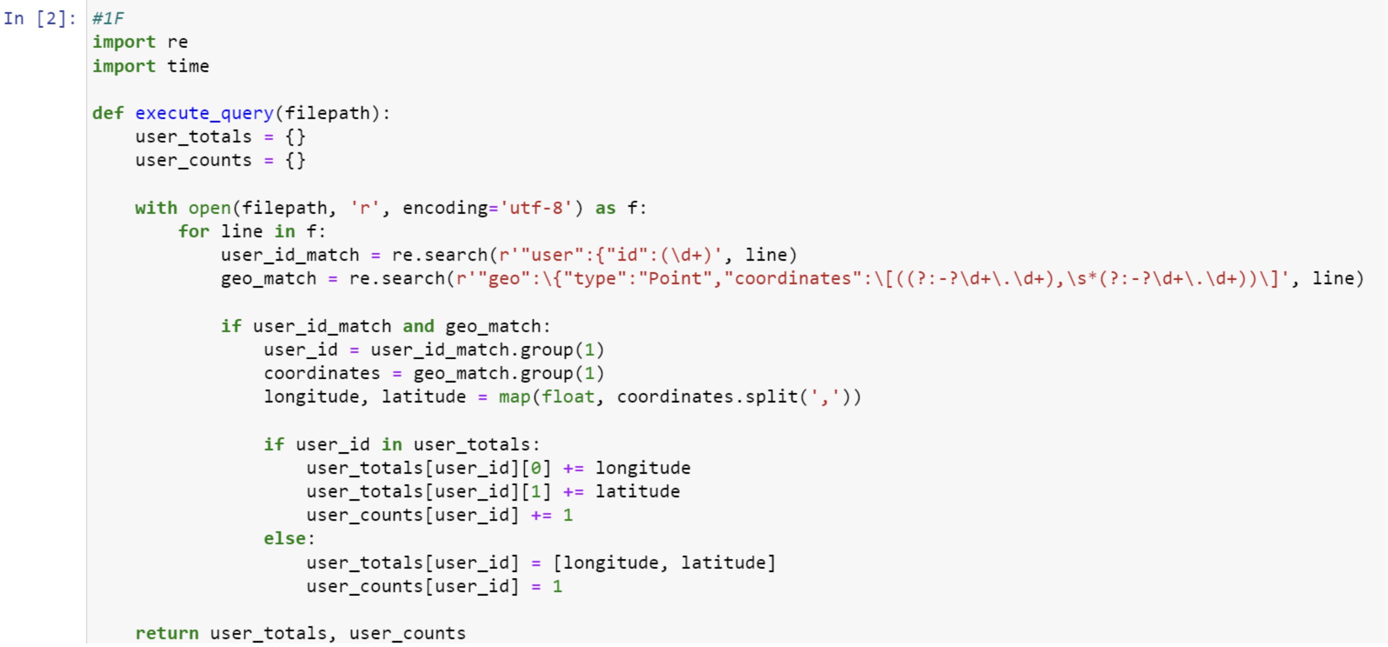
```
In [1]: import re
        import time

        user_totals = {}
        user_counts = {}

        filepath = 'C:/Users/nachi/Desktop/Databases/Final_BD/tweets_110000.txt'

        start = time.time()
        with open(filepath, 'r', encoding='utf-8') as f:
            for line in f:
                user_id_match = re.search(r'"user":{"id":(\d+)', line)
                geo_match = re.search(r'"geo":\{"type":"Point","coordinates":\[((?:-?\d+\.\d+),\s*(?:-?\d+\.\d+))\]', line)

                if user_id_match and geo_match:
                    user_id = user_id_match.group(1)
                    coordinates = geo_match.group(1)
                    longitude, latitude = map(float, coordinates.split(','))

                    if user_id in user_totals:
                        user_totals[user_id][0] += longitude
                        user_totals[user_id][1] += latitude
                        user_counts[user_id] += 1
                    else:
                        user_totals[user_id] = [longitude, latitude]
                        user_counts[user_id] = 1
```



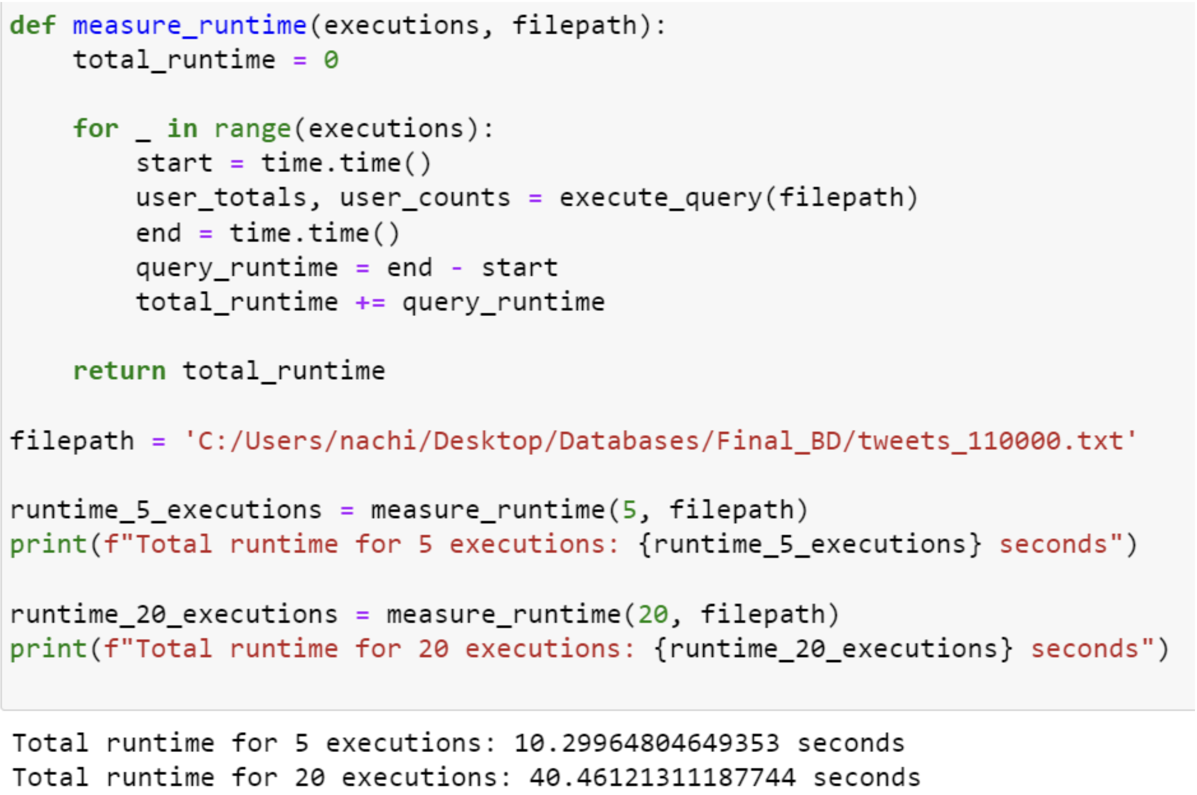
```
average_coordinates = {}
for user_id, totals in user_totals.items():
    count = user_counts[user_id]
    average_longitude = totals[0] / count
    average_latitude = totals[1] / count
    average_coordinates[user_id] = (average_longitude, average_latitude)
end = time.time()
query_runtime = end - start
print(f"Query runtime: {query_runtime} \n")
for user_id, coordinates in average_coordinates.items():
    print(f"User ID: {user_id}, Average Longitude: {coordinates[0]}, Average Latitude: {coordinates[1]}")
```

```
Query runtime: 2.0304901599884033

User ID: 160370249, Average Longitude: 14.670275, Average Latitude: 121.043955
User ID: 233079540, Average Longitude: -7.351872, Average Latitude: 110.213471
User ID: 146612119, Average Longitude: 47.8487, Average Latitude: -122.222
User ID: 348864517, Average Longitude: 38.767654, Average Latitude: -77.159617
User ID: 128825864, Average Longitude: -6.149429, Average Latitude: 106.728999
User ID: 254674397, Average Longitude: 52.536283, Average Latitude: -1.505078
User ID: 2328860924, Average Longitude: 26.198516, Average Latitude: 127.698844
User ID: 1022608538, Average Longitude: -29.716929, Average Latitude: -57.086503
User ID: 15610222, Average Longitude: 40.388261, Average Latitude: -75.273898
User ID: 500289098, Average Longitude: 36.109317, Average Latitude: -115.170286
User ID: 1560357476, Average Longitude: -23.953996, Average Latitude: -46.288466
User ID: 1631720540, Average Longitude: -29.188215, Average Latitude: -51.216072
User ID: 310891193, Average Longitude: 39.968613, Average Latitude: -75.278031
User ID: 610180858, Average Longitude: 34.771422, Average Latitude: -77.359981
User ID: 222471428, Average Longitude: -25.299878, Average Latitude: -57.612373
User ID: 215827490, Average Longitude: -22.678829, Average Latitude: -43.288543
User ID: 403383222, Average Longitude: 0.442102, Average Latitude: 101.451366
```

f. Re-execute the query in part 2-e 5 times and 20 times and measure the total runtime. Does the runtime scale linearly?



```
In [2]: #1F
        import re
        import time

        def execute_query(filepath):
            user_totals = {}
            user_counts = {}

            with open(filepath, 'r', encoding='utf-8') as f:
                for line in f:
                    user_id_match = re.search(r'"user":{"id":(\d+)', line)
                    geo_match = re.search(r'"geo":\{"type":"Point","coordinates":\[((?:-?\d+\.\d+),\s*(?:-?\d+\.\d+))\]', line)

                    if user_id_match and geo_match:
                        user_id = user_id_match.group(1)
                        coordinates = geo_match.group(1)
                        longitude, latitude = map(float, coordinates.split(','))

                        if user_id in user_totals:
                            user_totals[user_id][0] += longitude
                            user_totals[user_id][1] += latitude
                            user_counts[user_id] += 1
                        else:
                            user_totals[user_id] = [longitude, latitude]
                            user_counts[user_id] = 1

            return user_totals, user_counts
```



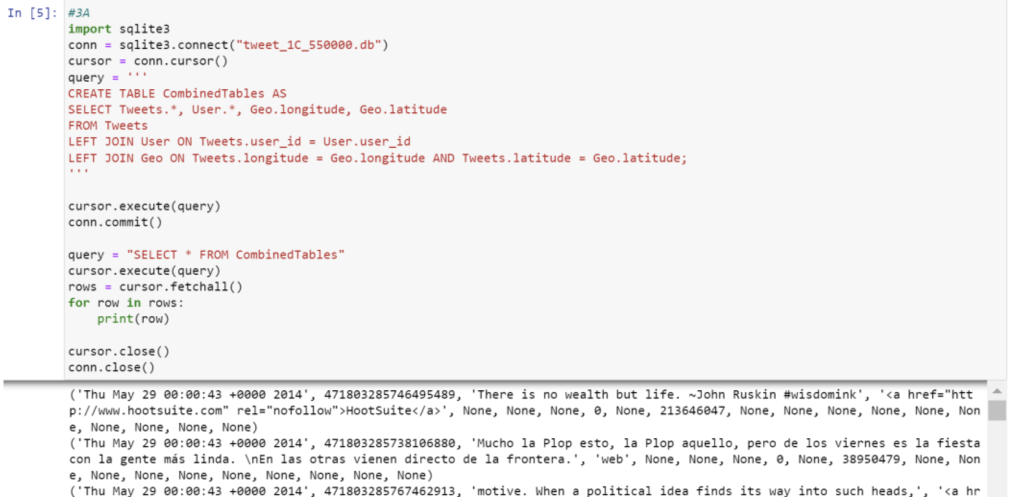
```
def measure_runtime(executions, filepath):
    total_runtime = 0

    for _ in range(executions):
        start = time.time()
        user_totals, user_counts = execute_query(filepath)
        end = time.time()
        query_runtime = end - start
        total_runtime += query_runtime

    return total_runtime

filepath = 'C:/Users/nachi/Desktop/Databases/Final_BD/tweets_110000.txt'

runtime_5_executions = measure_runtime(5, filepath)
print(f"Total runtime for 5 executions: {runtime_5_executions} seconds")

runtime_20_executions = measure_runtime(20, filepath)
print(f"Total runtime for 20 executions: {runtime_20_executions} seconds")
```

```
Total runtime for 5 executions: 10.29964804649353 seconds
Total runtime for 20 executions: 40.46121311187744 seconds
```

Answer 2f: Total Runtime (20 executions) / Total Runtime (5 executions) = 40.5 seconds / 10.3 seconds = 3.94. Which is approximately 4 we can say that the runtime scales almost linearly.

Part 3

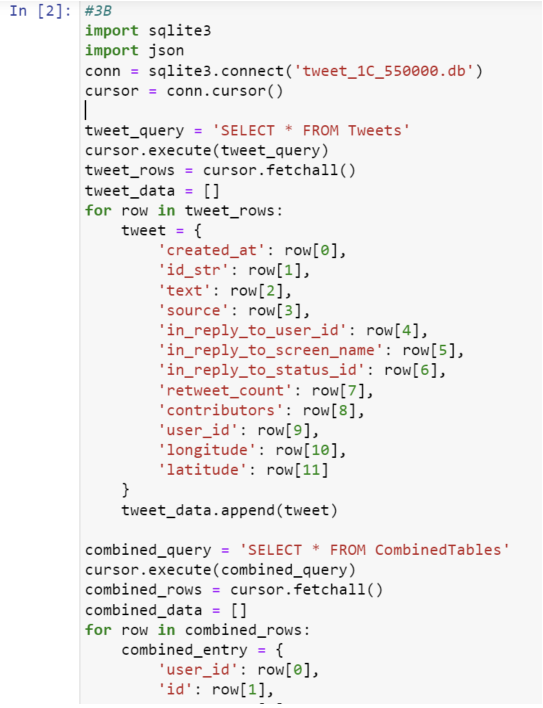
a. **Using the database with 550,000 tweets, create a new table that corresponds to the join of all 3 tables in your database, <u>including records without a geo location</u>. This is the equivalent of a materialized view but since SQLite does not support MVs, we will use CREATE TABLE AS SELECT (instead of CREATE MATERIALIZED VIEW AS SELECT).**



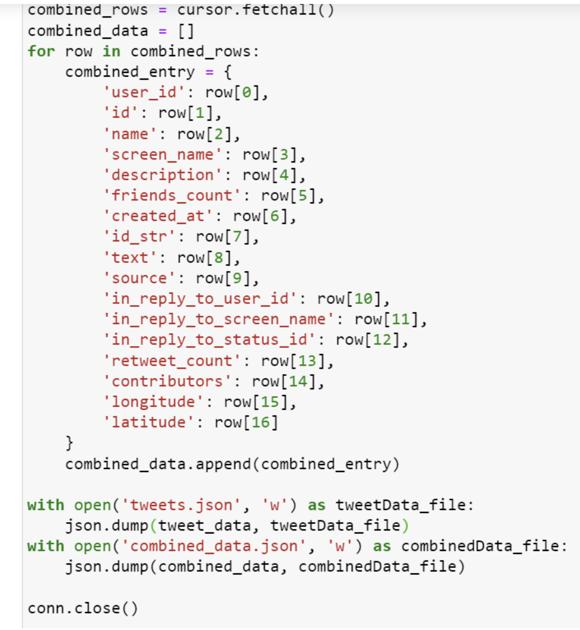
```
In [5]: #3A
        import sqlite3
        conn = sqlite3.connect("tweet_1C_550000.db")
        cursor = conn.cursor()
        query = '''
        CREATE TABLE CombinedTables AS
        SELECT Tweets.*, User.*, Geo.longitude, Geo.latitude
        FROM Tweets
        LEFT JOIN User ON Tweets.user_id = User.user_id
        LEFT JOIN Geo ON Tweets.longitude = Geo.longitude AND Tweets.latitude = Geo.latitude;
        '''

        cursor.execute(query)
        conn.commit()

        query = "SELECT * FROM CombinedTables"
        cursor.execute(query)
        rows = cursor.fetchall()
        for row in rows:
            print(row)

        cursor.close()
        conn.close()
```

```
('Thu May 29 00:00:43 +0000 2014', 471803285746495489, 'There is no wealth but life. ~John Ruskin #wisdomink', '<a href="htt
p://www.hootsuite.com" rel="nofollow">HootSuite</a>', None, None, None, 0, None, 213646047, None, None, None, None, None, Non
e, None, None, None, None)
('Thu May 29 00:00:43 +0000 2014', 471803285738106880, 'Mucho la Plop esto, la Plop aquello, pero de los viernes es la fiesta
con la gente más linda. \nEn las otras vienen directo de la frontera.', 'web', None, None, None, 0, None, 38950479, None, Non
e, None, None, None, None, None, None, None, None)
('Thu May 29 00:00:43 +0000 2014', 471803285767462913, 'motive. When a political idea finds its way into such heads,', '<a hr
```

b. Export the contents of 1) the Tweet table and 2) your new table from 3-a into a new JSON file (i.e., create your own JSON file with just the keys you extracted). You do not need to replicate the structure of the input and can come up with any reasonable keys for each field stored in JSON structure (e.g., you can have longitude as “longitude” key when the location is available).
How do the file sizes compare to the original input file?



```
In [2]: #3B
        import sqlite3
        import json
        conn = sqlite3.connect('tweet_1C_550000.db')
        cursor = conn.cursor()

        tweet_query = 'SELECT * FROM Tweets'
        cursor.execute(tweet_query)
        tweet_rows = cursor.fetchall()
        tweet_data = []
        for row in tweet_rows:
            tweet = {
                'created_at': row[0],
                'id_str': row[1],
                'text': row[2],
                'source': row[3],
                'in_reply_to_user_id': row[4],
                'in_reply_to_screen_name': row[5],
                'in_reply_to_status_id': row[6],
                'retweet_count': row[7],
                'contributors': row[8],
                'user_id': row[9],
                'longitude': row[10],
                'latitude': row[11]
            }
            tweet_data.append(tweet)

        combined_query = 'SELECT * FROM CombinedTables'
        cursor.execute(combined_query)
        combined_rows = cursor.fetchall()
        combined_data = []
        for row in combined_rows:
            combined_entry = {
                'user_id': row[0],
                'id': row[1],
```

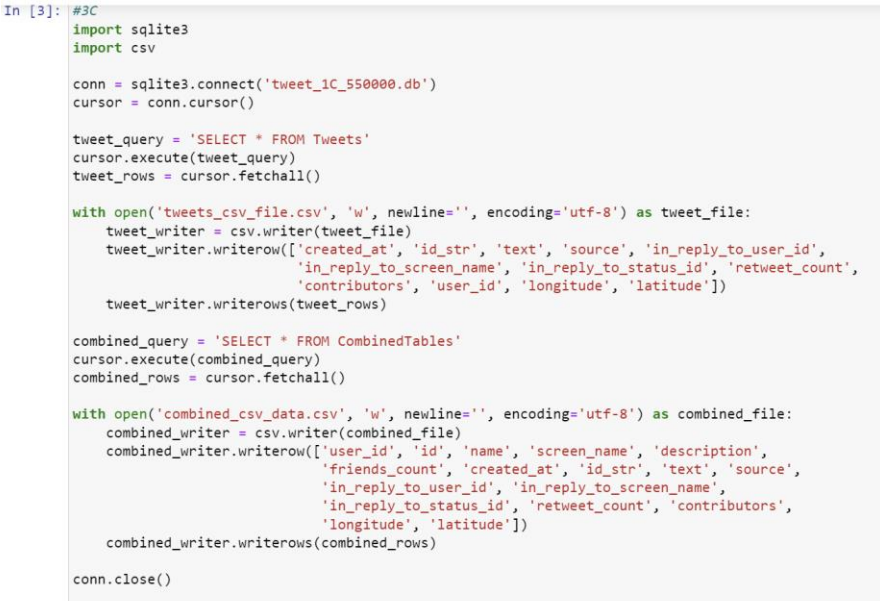


```
combined_rows = cursor.fetchall()
combined_data = []
for row in combined_rows:
    combined_entry = {
        'user_id': row[0],
        'id': row[1],
        'name': row[2],
        'screen_name': row[3],
        'description': row[4],
        'friends_count': row[5],
        'created_at': row[6],
        'id_str': row[7],
        'text': row[8],
        'source': row[9],
        'in_reply_to_user_id': row[10],
        'in_reply_to_screen_name': row[11],
        'in_reply_to_status_id': row[12],
        'retweet_count': row[13],
        'contributors': row[14],
        'longitude': row[15],
        'latitude': row[16]
    }
    combined_data.append(combined_entry)

with open('tweets.json', 'w') as tweetData_file:
    json.dump(tweet_data, tweetData_file)
with open('combined_data.json', 'w') as combinedData_file:
    json.dump(combined_data, combinedData_file)

conn.close()
```

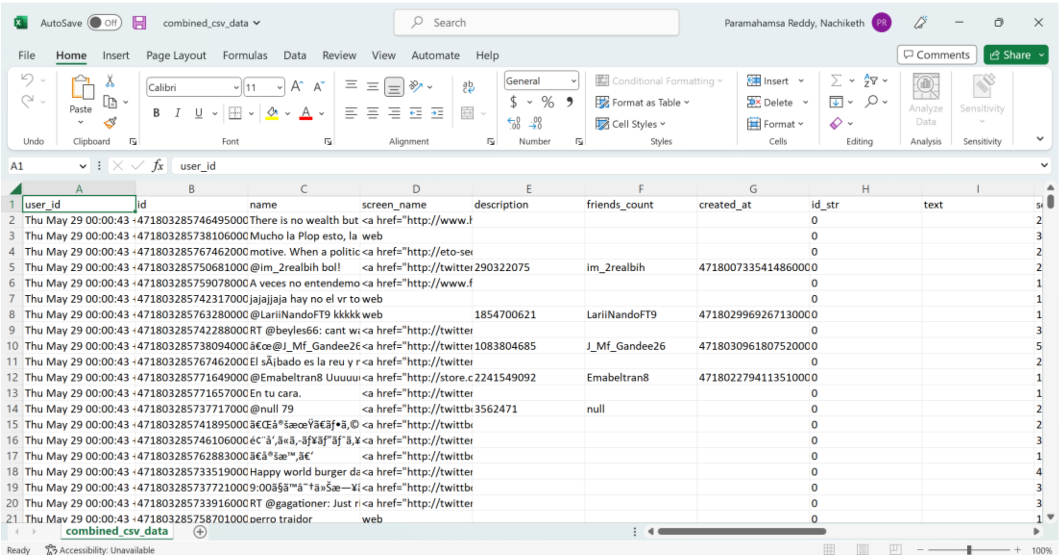
ORIGINAL FILE SIZE: 1.69 gb (1,696,661 kb)
TWEETS JSON FILE SIZE: 26.5 mb (265,462 kb)
COMBINED DATA JSON FILE SIZE: 31.4 mb (314,343 kb)

The original file size is very large compared to the JSON files. We can see that data is stored at an optimized level. This reduction in file size implies that the data is stored in the JSON format in an optimal manner. JSON files efficiently reflect the data structure and remove extraneous data, resulting in a smaller file size without affecting data integrity.

c. **Export the contents of 1) the Tweet table and 2) your table from 3-a into a .csv (comma separated value) file. How do the file size compare to the original input file and to the files in 3-b?**

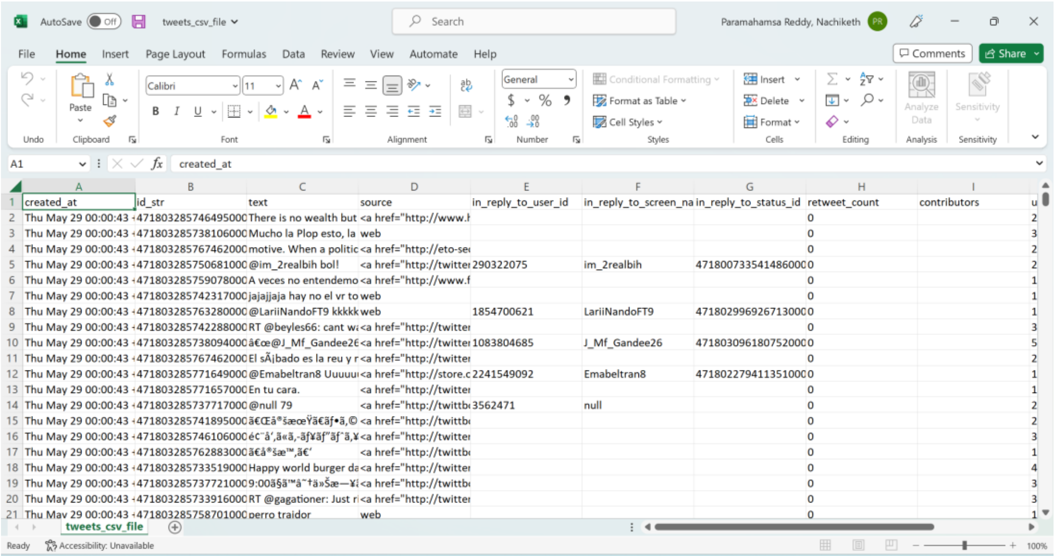


```
In [3]: #3C
import sqlite3
import csv

conn = sqlite3.connect('tweet_1C_550000.db')
cursor = conn.cursor()

tweet_query = 'SELECT * FROM Tweets'
cursor.execute(tweet_query)
tweet_rows = cursor.fetchall()

with open('tweets_csv_file.csv', 'w', newline='', encoding='utf-8') as tweet_file:
    tweet_writer = csv.writer(tweet_file)
    tweet_writer.writerow(['created_at', 'id_str', 'text', 'source', 'in_reply_to_user_id',
                           'in_reply_to_screen_name', 'in_reply_to_status_id', 'retweet_count',
                           'contributors', 'user_id', 'longitude', 'latitude'])
    tweet_writer.writerows(tweet_rows)

combined_query = 'SELECT * FROM CombinedTables'
cursor.execute(combined_query)
combined_rows = cursor.fetchall()

with open('combined_csv_data.csv', 'w', newline='', encoding='utf-8') as combined_file:
    combined_writer = csv.writer(combined_file)
    combined_writer.writerow(['user_id', 'id', 'name', 'screen_name', 'description',
                              'friends_count', 'created_at', 'id_str', 'text', 'source',
                              'in_reply_to_user_id', 'in_reply_to_screen_name',
                              'in_reply_to_status_id', 'retweet_count', 'contributors',
                              'longitude', 'latitude'])
    combined_writer.writerows(combined_rows)

conn.close()
```



	A	B	C	D	E	F	G	H	I	
1	user_id	id	name	screen_name	description	friends_count	created_at	id_str	text	s
2	Thu May 29 00:00:43	471803285746495000	There is no wealth but	<a href="http://www.h				0		2
3	Thu May 29 00:00:43	471803285738106000	Mucho la Plop esto, la	web				0		3
4	Thu May 29 00:00:43	471803285767462000	motive. When a politic	<a href="http://eto-se				0		2
5	Thu May 29 00:00:43	471803285750681000	@im_2realbih bol!	<a href="http://twitter	290322075	im_2realbih	471800733541486000	0		2
6	Thu May 29 00:00:43	471803285759078000	A veces no entendemo	<a href="http://www.f				0		1
7	Thu May 29 00:00:43	471803285742317000	jajajjaja hay no el vr to	web				0		1
8	Thu May 29 00:00:43	471803285763280000	@LariiNandoFT9 kkkkk	web	1854700621	LariiNandoFT9	471802996926713000	0		1
9	Thu May 29 00:00:43	471803285742288000	RT @beyles66: cant wa	<a href="http://twitter				0		3
10	Thu May 29 00:00:43	471803285738094000	â€œ@J_Mf_Gandee26	<a href="http://twitter	1083804685	J_Mf_Gandee26	471803096180752000	0		5
11	Thu May 29 00:00:43	471803285767462000	El sÃ¡bado es la reu y r	<a href="http://twitter				0		2
12	Thu May 29 00:00:43	471803285771649000	@Emabeltran8 Uuuuu	<a href="http://store.c	2241549092	Emabeltran8	471802279411351000	0		1
13	Thu May 29 00:00:43	471803285771657000	En tu cara.	<a href="http://twitter				0		1
14	Thu May 29 00:00:43	471803285737717000	@null 79	<a href="http://twittb	3562471	null		0		2
15	Thu May 29 00:00:43	471803285741895000	ã€Œå®šæœŸã€ãƒ•ã,©	<a href="http://twittb				0		2
16	Thu May 29 00:00:43	471803285746106000	é¢¨å',ã«ã,-ãƒ¥ãƒ"ãƒˆã,¥	<a href="http://twitter				0		3
17	Thu May 29 00:00:43	471803285762883000	ã€å®šæ™,ã€'	<a href="http://twittb				0		1
18	Thu May 29 00:00:43	471803285733519000	Happy world burger da	<a href="http://twitter				0		4
19	Thu May 29 00:00:43	471803285737721000	9:00ã§ã™â˜†ä»Šæ—¥ã	<a href="http://twittb				0		3
20	Thu May 29 00:00:43	471803285733916000	RT @gagationer: Just ri	<a href="http://twitter				0		3
21	Thu May 29 00:00:43	471803285758701000	perro traidor	web				0		1

combined_csv_data



created_at

	A	B	C	D	E	F	G	H	I	
1	created_at	id_str	text	source	in_reply_to_user_id	in_reply_to_screen_na	in_reply_to_status_id	retweet_count	contributors	u
2	Thu May 29 00:00:43 +	47180328574649500	There is no wealth but	<a href="http://www.h				0		2
3	Thu May 29 00:00:43 +	47180328573810600	Mucho la Plop esto, la	web				0		3
4	Thu May 29 00:00:43 +	47180328576746200	motive. When a politic	<a href="http://eto-se				0		2
5	Thu May 29 00:00:43 +	47180328575068100	@im_2realbih bol!	<a href="http://twitter	290322075	im_2realbih	47180073354148600	0		2
6	Thu May 29 00:00:43 +	47180328575907800	A veces no entendemo	<a href="http://www.f				0		1
7	Thu May 29 00:00:43 +	47180328574231700	jajajjaja hay no el vr to	web				0		1
8	Thu May 29 00:00:43 +	47180328576328000	@LariiNandoFT9 kkkkk	web	1854700621	LariiNandoFT9	47180299692671300	0		1
9	Thu May 29 00:00:43 +	47180328574228800	RT @beyles66: cant wa	<a href="http://twitter				0		3
10	Thu May 29 00:00:43 +	47180328573809400	â€œ@J_Mf_Gandee26	<a href="http://twitter	1083804685	J_Mf_Gandee26	47180309618075200	0		5
11	Thu May 29 00:00:43 +	47180328576746200	El sÃ¡bado es la reu y r	<a href="http://twitter				0		2
12	Thu May 29 00:00:43 +	47180328577164900	@Emabeltran8 Uuuuu	<a href="http://store.c	2241549092	Emabeltran8	47180227941135100	0		1
13	Thu May 29 00:00:43 +	47180328577165700	En tu cara.	<a href="http://twitter				0		1
14	Thu May 29 00:00:43 +	47180328573771700	@null 79	<a href="http://twittb	3562471	null		0		2
15	Thu May 29 00:00:43 +	47180328574189500	ã€Œå®šæœŸã€ãƒ•ã‚©	<a href="http://twittb				0		2
16	Thu May 29 00:00:43 +	47180328574610600	é€¨å',ã«ã,-ãƒ¥ãƒ"ãƒˆã,¥	<a href="http://twitter				0		3
17	Thu May 29 00:00:43 +	47180328576288300	ã€å®šæ™,ã€'	<a href="http://twittb				0		1
18	Thu May 29 00:00:43 +	47180328573351900	Happy world burger da	<a href="http://twitter				0		4
19	Thu May 29 00:00:43 +	47180328573772100	9:00ã§ã™â˜†ä»Šæ—¥ã	<a href="http://twittb				0		3
20	Thu May 29 00:00:43 +	47180328573391600	RT @gagationer: Just r	<a href="http://twitter				0		3
21	Thu May 29 00:00:43 +	47180328575870100	perro traidor	web				0		1

tweets_csv_file

TWEETS CSV FILE SIZE: 12.7 mb (127,079kb)
COMBINED DATA CSV FILE SIZE: 13.1 mb (131,623kb)
CSV files contain data in tabular format, with each line representing a row and values separated by commas. Because this layout avoids the need for repetitive field names and other metadata, the file size is reduced.